Eric Horlait
Thomas Magedanz
Roch H. Glitho (Eds.)

LNCS 2881

Mobile Agents for Telecommunication Applications

5th International Workshop, MATA 2003
Marrakech, Morocco, October 2003
Proceedings

Springer

Lecture Notes in Computer Science 2881
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Eric Horlait   Thomas Magedanz
Roch H. Glitho (Eds.)

Mobile Agents for Telecommunication Applications

5th International Workshop, MATA 2003
Marrakech, Morocco, October 8-10, 2003
Proceedings

Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editor

Eric Horlait
Pierre and Marie Curie University
LIP6 Laboratory
4, place Jussieu, 75252 Paris Cedex 05, France
E-mail: Eric.Horlait@lip6.fr

Thomas Magedanz
Technische Universität Berlin, FhG FOKUS
Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany
E-mail: tm@cs.tu-berlin.de

Roch H. Glitho
Ericsson Research Canada, 8400 Decarie Blvd.
Town of Mount Royal, Quebec, Canada H4P 2N2
E-mail: roch.glitho@ericsson.ca

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

CR Subject Classification (1998): C.2, D.2.12, I.2.11, D.4.3-4, D.4.6, E.1, H.3

ISSN 0302-9743
ISBN 3-540-20298-6 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by PTP-Berlin GmbH
Printed on acid-free paper SPIN 10966037 06/3142 5 4 3 2 1 0

Preface

The aim of the MATA workshops series is to provide a unique opportunity for researchers from the IT, Internet, and telecommunications domain, as well as related software and application developers and service providers to discuss the advances in agent technologies and their applications in next generation mobile Internet and telecommunications.

Since 1999 in Canada, MATA workshops have contributed to the creation of a research community around mobile agents and their use in telecommunication applications.

The 2003 workshop focused on recent developments in agent technologies and particularly the use of agent technologies within the fields of network management, dynamic service provisioning and management, nomadic and mobile computing, context aware services and environments, active and programmable networks, policybased services and management, ad hoc networking, peer-to-peer computing, ambient intelligence, Wireless Java, software defined radio, adaptive mobile end systems, virtual home environments, smart home, smart cars and navigation, e-learning, m-commerce, and other related 3Gb areas.

October 2003 Eric HORLAIT

Scientific Program Committee

Table of Contents

Efficient Formation of Dynamic Bluetooth Scatternet via Mobile Agent Processing

Sergio Gonzalez-Valenzuela[1] , Son T. Vuong[2] and Victor C.M. Leung[1]

[1] Department of Electrical and Computer Engineering
[2] Department of Computer Science
The University of British Columbia
Vancouver, B.C. Canada V6T 1Z4
{sergiog,vleung}@ece.ubc.ca and vuong@cs.ubc.ca

Abstract. During recent years, there has been an increasing interest in the research and development of Wireless Personal Area Networks (WPANs). In particular, Bluetooth has been widely acknowledged as a suitable solution to enable small electronic devices with wireless connectivity. However, a number of issues remain to be solved before Bluetooth-equipped devices can provide users with full ad-hoc networking capabilities. This paper explores and discusses the applicability of mobile agent technology to address a specific issue in Bluetooth networks: the scatternet formation problem. It is argued that mobile agent technology can be efficiently employed to solve this particular problem in Bluetooth WPANs, overcoming restrictions seen in existing schemes. Simulations results demonstrate the effectiveness of using mobile processing to solve the scatternet formation problem.

Keywords: Mobile agents, mobile processing, Bluetooth, scatternet formation, ad-hoc networks, and wireless personal area networks.

1 Introduction

Providing small computing devices with wireless connectivity is one of the key steps toward enabling mobile users with ubiquitous computing capability. To this effect, Bluetooth technology has been widely adopted as an alternative solution to other wireless technologies. Its simplified radio circuitry is specifically tailored to meet the requirements of small electronic gadgets, such as cellular phones and personal digital assistants. Its current specification provides wireless connectivity of up to 10 metres, and a maximum data rate of 721 Kbps [1]. Wider coverage areas are also possible, e.g. up to 100 metres, while additional specifications are being developed to reach data rates as high as 55 Mbps [2]. Given these features, and a target price of a few dollars, Bluetooth is widely considered a promising technology to enable WPANs [3], [4]. However, before this possibility becomes a reality, a number of issues inherent to the Bluetooth wireless architecture must be addressed. One particular problem is the need for efficient solutions to configure ***scatternets***, a term coined to Bluetooth-enabled ad-hoc networks. ***Scatternets*** possess unique characteristics that are inherent to the Bluetooth technology, which adds extra complexity to scatternet formation protocols. This paper presents a novel solution, in which mobile (agent) processing is employed to dynamically configure scatternets. To our knowledge, there has been no existing work reported in the literature, that applies agents technology and mobile processing to the problem of dynamic Bluetooth scatternet formation. Not including this introduction section, this paper is organized into 6 sections as follows. Section 2 reviews the basics of Bluetooth technology to give a foundation for the rest of the paper. Section 3 gives a brief description of the mobile-processing paradigm and its applicability to ad-hoc networking. Section 4 elaborates on the proposed method to solve the scatternet formation problem using mobile processing. Simulation results are presented in Section 5 to demonstrate the effectiveness of the proposed method. Finally, in Section 6 we offers concluding remarks and suggestions for future work.

2 Introduction to Bluetooth WPANs

In comparison to the wireless LAN technology, establishing a communication link between two Bluetooth enabled devices is a somewhat more intricate process. This can be primarily attributed to the mechanism used in the Bluetooth architecture for devices to gain access to the wireless channel. Such mechanism is known as Frequency-Hopping Spread Spectrum (FH-SS), which is widely acknowledged to provide high bandwidth utilization and improved noise immunity. Bluetooth devices (BDs) establish a communications link by synchronizing the pseudo-random sequence with which they sequentially access different channels in the available bandwidth according to the FH-SS mechanism [1]. To accomplish this, BDs must first become aware of their mutual existence. This is the primary objective during the first stage of the discovery procedure, known as the *inquiry* process. Once two BDs have become aware of their mutual existence, both devices are able to complete the synchronization procedure by entering the *page* state. This second stage enables one device to transfer hardware-specific data to the other, leading to the desired synchronization that precedes the *connected* state. A drawback to this scheme is that the delay incurred during the inquiry process can be as long as 10 seconds [1].

According to the current architecture of the Bluetooth technology, after a wireless link has been created between two BDs, one of the devices assumes the role of a *master*, while the other assumes the role of a *slave.* More than two BDs can connect to create a ***piconet***. In a ***piconet***, there can be only one master, and up to seven active slaves. Dual master-slave roles in a BD are also possible, but are rather avoided due to further role-swapping delays. A master BD has the task of controlling the data flow among the members of its piconet that inherently has a star topology centred at the master BD. A piconet can accommodate additional slaves as long as they assume a *parking* (inactive) role. A scheduling scheme can then be enforced to control the way in which slaves swap between *connected* or *parking* states to enable data transfer as required. In addition, a slave BD may belong to more than one piconet by assuming an active role in one of them and a parking role on the others in a rotating fashion. Such a BD, referred as a bridge, can then be used to relay data among piconets, resulting in the scatternet concept shown in Figure 1.

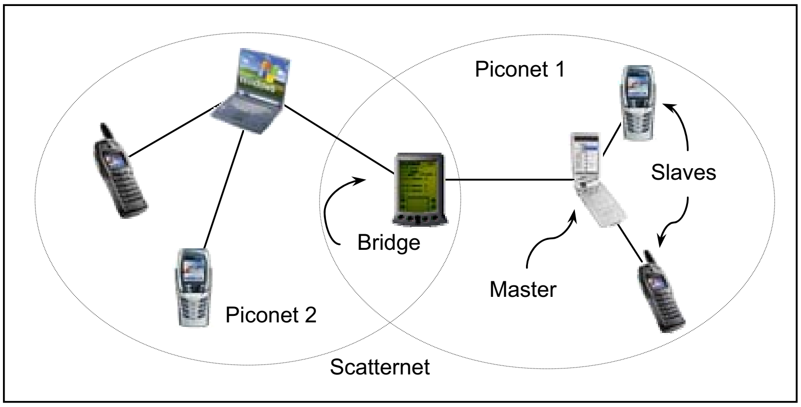


Fig. 1. A sample scatternet

Given the renewed interest by the telecommunications industry [4], an increasing number of researchers in the area have put forward proposals and investigation reports that deal with the scatternet formation problem. The scatternet formation problem can be informally stated as follows. Given a set of BDs, how can they be interconnected to best form a set of piconets that constitute an optimal scatternet? Additionally, how can a new BD be optimally connected to an existing scatternet? An optimal scatternet topology can be defined as the one that has the minimal number of piconets (or *masters,* i.e. the fewest average number of hops required between any pair of BDs. Rather than describing these existing approaches, it is important to note the assumptions behind these protocols that explain their limitations. For instance,

- BDs participating in a scatternet formation process are somehow able to initialize the protocol at roughly the same time [5], [6];
- all BDs must be within radio range of each other for the protocol to work [7], [8];
- additional BDs cannot join the scatternet at a later time [9].

It is not difficult to notice that these assumptions may not necessarily hold in practice when the Bluetooth technology is applied to support WPANs. It is therefore imperative that scatternet formation protocols be able to overcome these limitations, and adhere to the needs of the users. The next section presents some necessary background on the mobile agent paradigm as an alternative approach to the scatter network formation problem, that overcomes the limitations.

3 Applicability of Mobile Agents in Bluetooth WPANs

The applicability of the mobile agent paradigm to solve diverse networking problems has been broadly studied. Due to the dynamic nature of ad-hoc networks, the potential usefulness of mobile agent systems in this area seems natural. In fact, the incursion of mobile agent schemes into the wireless networks realm has already taken place with successful results, particularly in the routing area [10], [11]. To the best of the authors' knowledge, this is the first time that a mobile agent system is employed to address the scatternet formation problem.

Within the software context, mobile agents may be defined as self-contained units of code that can be moved across computer networks to perform a specific task on behalf of the entity that uses it. Such entity may be either a human user or a computing system. According to existing concepts, mobile agents should possess certain characteristics, such as compactness, autonomy and persistence [12]. The use of mobile agents is generally targeted at the application layer for brokering-like tasks. However, a number of systems have used this technology to address other network-level issues. In the later case, the use of mobile agent 'colonies' is a common method, where such a system is comprised of not one, but potentially many agents collaborating to achieve a given task. This same framework is employed in the novel solution we introduce in this paper.

Several mobile agent systems have been investigated and developed by the research community worldwide. Most of the existing systems make use of the Java language to implement mobile agents. The *Wave* system has alternatively been used in this and other investigations [13], [14]. Although regarded as a mobile agent system, Wave can also be viewed as a mobile processing software tool that encompasses a number of characteristics not seen in other systems [15]. In particular, Wave incorporates the notion of both a track layer and a knowledge-network layer. These mechanisms enable a robust processing architecture that supports strong migration, among other features. As a consequence, a number of intricacies that are usually the responsibility of the programmer can be instead managed by the Wave Interpreter (WI), leading to a more compact code and improved functionality [16]. These latter aspects of the Wave mobile processing system are of prime importance when considering alternative approaches in an attempt to reduce the amount of network management traffic normally seen in highly dynamic ad-hoc networks [17]. Obviously, network management traffic is an overhead that should be minimized as much as possible in such bandwidth-limited environments.

Thus, a light-weight Wave-like system running on small wireless devices would be highly desirable in enabling them to achieve a high degree of coordination among themselves. However, before this step is taken, it is necessary to first investigate the suitability of using mobile agents to solve the dynamic ad-hoc network formation problem at the simulation level. We explain how the scatternet formation problem is explored from a mobile processing point of view in the next section.

4 Scatternet Formation through Mobile Processing

Our simulation of scatternet formation by mobile processing is based on a few assumptions. First, we assume that a routing/forwarding mechanism is already in place at every BD participating in the protocol; at the very least, BDs should possess some basic forwarding mechanism to relay data to neighbouring BDs. Another assumption is that mobile agents (Wave agents) must have access to the Host Controller Interface (HCI) in every BD in order to carry out the actual scatternet configuration. Such Wave agents can then make use of an Application Programming Interface (API) embedded into the BDs' middleware, ridding the need for the agent to carry additional code. In addition, a newly discovered BD does not need to be within radio range of every other BD in a pre-existing scatternet. This implies that on-demand (dynamic) scatternet configuration is supported. For now, BD mobility is not considered. Finally, our approach results in the dynamic configuration of scatternets that possess a tree topology.

A *master* node is potentially the busiest one in a piconet due to its inherent traffic controlling role, leaving little time to spare for other tasks. On the other hand, the period of time in which slave nodes wait for their scheduled access to the wireless channel can be used to discover new BDs. Thus, it is reasonable to expect that such newly arrived BDs would more likely be discovered by slave nodes in a given piconet. A similar consideration was pointed out in [18]. In such case, newly discovered BDs would have to assume a master role, leading to the almost uncontrolled creation of a new piconet every time a new BD is discovered. This situation is undesirable, as a higher number of piconets in a scatternet implies a larger number of hops (and thus a longer data transfer delay), as well as increased cluttering of the medium's bandwidth [3].

This is where the mobile processing solution comes into play. We start with a pre-existing piconet with at least two BDs. If a newly connected BD is discovered by a slave in this piconet, a mobile-agent-based reconfiguration process can be initiated to attempt topology optimization. Two specific circumstances are graphically depicted in Figure 2 where newly arrived BDs are shown as un-shaded masters. The one on the left hand is just within radio range of the existing master in Piconet 1, so a Wave agent is dispatched to alert the master of the newly discovered BD.

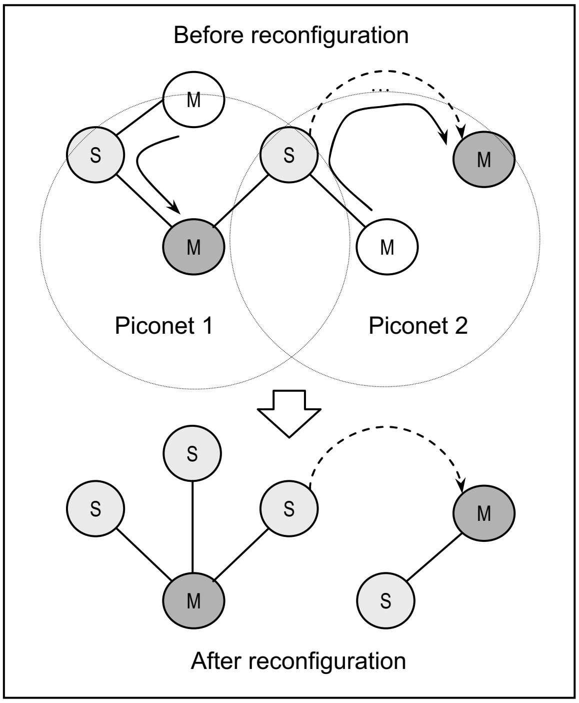


Fig. 2. On-demand scatternet configuration

In the second case, the new BD on the right hand is not within radio range of the master in Piconet 1, but within reach of the existing master in Piconet 2. The dashed arrow indicates however that there's no direct connection to such master; thus, Wave agents have to explore a longer route to reach the newly arrived BD in order to accomplish their objective.

In order for the previous scheme to work, Wave agents would have to carry the address of the newly arrived BD, along with the current value of its internal clock. Such information is sufficient for another BD to compute the current frequency hopping sequence of the new BD, allowing the targeted master node to synchronize with it, and thus enabling a subsequent paging process before entering the connected state [1].

The Wave system facilitates the implementation of a number of agent searching techniques that the user can employ to fit the needs of the system under development [15]. Of such techniques, an adequate one would be required to configure the scatternet when the second situation arises as explained in the previous example. For this first implementation, a breadth-first parallel-spread (BF-SS) searching technique is employed, in which agents coordinate a gradual breadth-first spreading of the mobile process. The difference between this modified scheme and the regular bread-first technique is that only one node is allowed to be active at a given time for every tier being probed, as described next.

The reconfiguration process is first attempted at the master of the slave that discovered the new BD. This can be accomplished by launching a Wave agent carrying the previously mentioned information about the new BD. A simple two-hop mobile agent forwarding is thus required before carrying out this initial probing, as shown in Piconet 1 of Figure 3. An unsuccessful probe causes the Wave agent to clone as many copies of itself as neighbouring piconets exist. The clone agents are then asynchronously forwarded to the masters located at subsequent tiers from the current location through (slave) bridge nodes. Once there, each individual clone agent signals its arrival to the same WI that launched it, which in turn acknowledges each agent's arrival in a sequential fashion. This operation is performed by means of the track layer of the Wave system. In this regard, the WI will automatically schedule an appropriate signalling scheme, depending on the way in which the agent's code has been structured. No explicit signalling command is required in the agent's code. Every agent is thus sequentially allowed to continue with its execution thread as scheduled by the preceding WI in the process. The actual radio proximity probing of the new BD can then be performed by computing its current hopping sequence using the BD-related information that each agent carries. The agent may then request access to the radio channel through the BD's HCI to enter the page state and perform the probing. When finished, the agent reports back the termination status of the probing process to the preceding WI.

Depending on the result obtained, the WI decides whether or not to continue with the process. An unsuccessful probe triggers a signal to the next scheduled agent. This enables every agent to continue in an identical manner as the previous one. The mobile process is recursively executed at subsequent piconet tiers, should all individual agents at the current tier return unsuccessful termination signals. The process stops upon finding a master who is able to both successfully page the newly arrived BD, and accommodate space within its piconet. When that happens, the new BD performs a master-to-slave role swap. The initial link between the new BD and the slave node that discovered it is finally broken, and a new link is setup to connect as a slave with the new master.

If no suitable master is found, a failure signal is propagated backwards to the original node that launched the first Wave agent. In this case, the whole process fails and the new BD keeps its master role. Clearly, employing a BF-SS technique ensures that no more than one master at a time will page the new BD, avoiding collisions on the medium. This is achieved by means of the enhanced Wave architecture that enables a superior coordination capability across a distributed network of Wave Interpreters. In addition, the BF-SS provides a bounded delay of the process completion time [19]. For details on the implementation of the BF-SS refer to [15].

5 Results

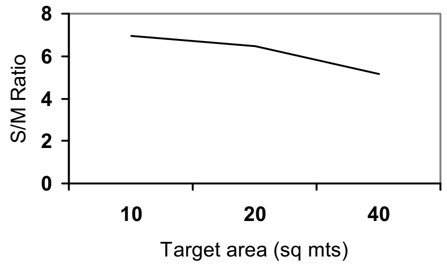
The scatternet formation scheme evaluated in this paper is based on an experimental Wave system available from [20]. Simulations comprise trial runs of 200 nodes arriving at uniformly distributed coordinates within test areas of 10, 20 and 40 square metres. For simplicity, the arrival of BD nodes is kept constant at rate of one per second, but other arrival random processes can be considered as well (say, Poisson).

Figure 3 shows the average results obtained for each of the areas being tested in the simulation. It can be observed that the target slave/master ratio achieved approaches a value of 7, the maximum targeted value (number) of active slaves per master in a given piconet. We also observe that such value degrades with the increasing size of the test area.

In addition, Figure 4 shows actual snapshots where test runs of 50 nodes were conducted. Figure 4a depicts a no-reconfiguration scheme, so that a newly arrived BD retains its original mode. An approximate 1:1 (26/24) slave/master ratio is achieved by such a scheme due to the randomness of the process.

Figure 4b shows a preliminary reconfiguration scheme where only the first tier from the current piconet is targeted. Visual inspection shows some noticeable improvement in the slave/master ratio (35/15= 2.33).

Finally, Figure 4c shows an n^{th}-tier reconfiguration scheme enabled by the BF-SS searching technique, where a clear improvement in the slave/master ratio can be observed (39/11 = 3.55).



Scatternet Size (m^2)	M/S Ratio
10 X 10	6.93
20 X 20	6.46
40 X 40	5.17

Fig. 3. Slave/Master ratio results

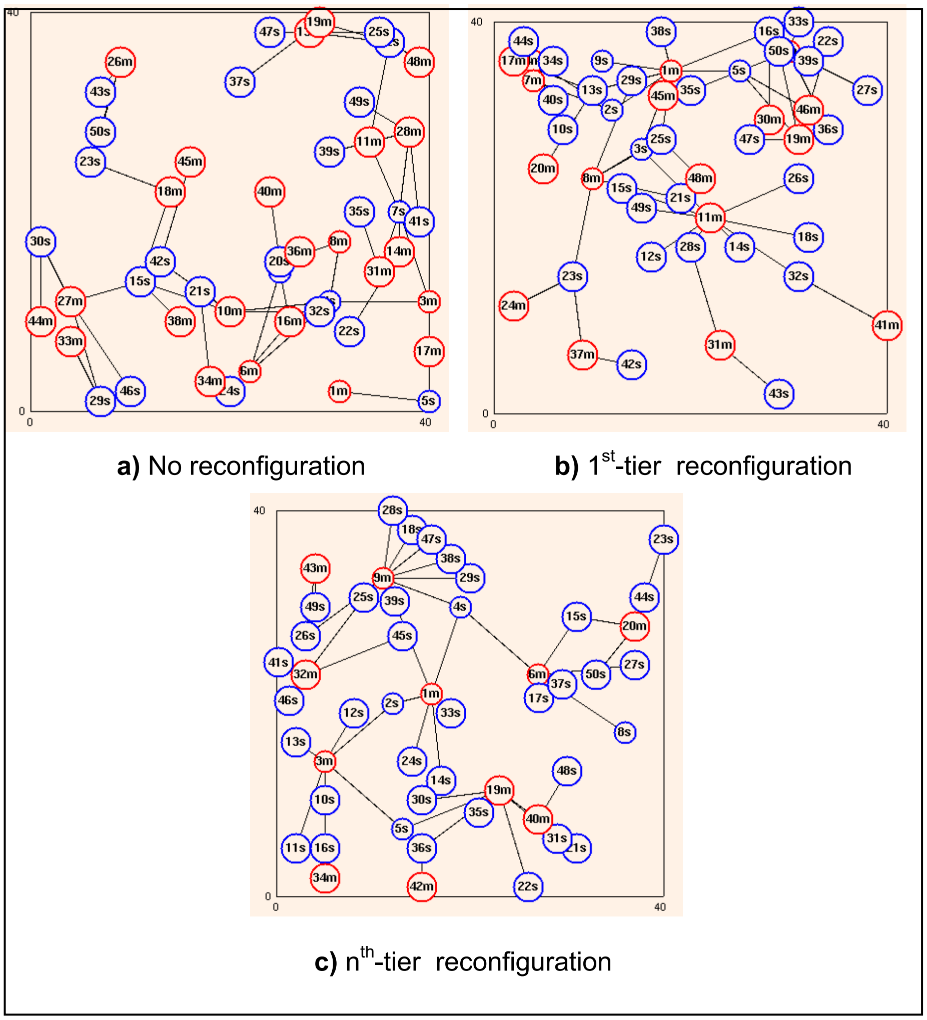


Fig. 4. Scatternet reconfiguration schemes

6 Conclusions

This paper proposes a mobile agent based approach to the problem of scatternet formation in Bluetooth WPANs . In this context, a protocol that satisfies the basic requirements to achieve the scatternet formation objective is presented. Preliminary results has demonstrated the viability and effectiveness of the novel mobile agent based approach to the configuration of this particular type of dynamic ad-hoc networks. The Wave mobile agent system was adopted and successfully employed for this purpose owing to its salient features that facilitate rapid prototyping for evaluation of network management systems. Other mobile agent platforms are expected be applicable as well.

We present a preliminary investigation of the benefits and challenges faced by application of a mobile processing platform to the problem of Bluetooth scatternet formation. Both interoperability issues and performance analysis still remain for future work as part of a comprehensive study on the application of mobile agent schemes within the wireless networking realm. The experimental version of the Wave Interpreter used here is currently suitable for PCs and workstations running Linux or Solaris. A ‘light-weight’ version of Wave would have to be implemented in order to fit the capabilities of wireless devices that possess reduced processing power. The implementation of such a customized system would enable other mobile agent-based network management tasks, such as routing or service discovery, in wireless adhoc networks.

Acknowledgements

This research is supported in parts by the National Sciences and Engineering Research Council of Canada (NSERC), the Canadian Institute for Telecommunications Research (CITR) Network of Centres of Excellence, and the Advanced Systems Institute (ASI) of British Columbia.

References

1. Bluetooth Specification Version 1.1, http://www.bluetooth.com, February 22, 2001.
2. IEEE 802.15 Working Group for WPANs, http://grouper.ieee.org/groups/802/15/.
3. Johansson, P. et. al. *“Bluetooth an Enabler of Personal Area Networking.”* IEEE Network, Special Issue in Personal Area Networks, September 2001.
4. Schneiderman, R. “Bluetooth’s Slow Dawn”, *IEEE Spectrum Online*, January 2001.
5. S. Basagni and C. Petrioli. “A Scatternet Formation Protocol For Ad Hoc Networks of Bluetooth Devices.” *In Proceedings of the IEEE Semi-annual Vehicular Technology Conference*, VTC May, 2002, Birmingham, AL, USA.
6. Wang, Z., Thomas, R. and Haas, Z. “Bluenet – A New Scatternet Formation Scheme.” *35th Annual Hawaii International Conference on System Sciences (HICSS'02)*, January 07 - 10, 2002 Big Island, Hawaii.
7. T. Salonidis, et. al., “Distributed Topology Construction of Bluetooth Personal Area Networks.” *In Proc. IEEE INFOCOM*, Anchorage, AK, April 2001.

8. C. Law, A. K. Mehta, and K.-Y. Siu. “Performance of a New Bluetooth Scatternet Formation Protocol”. *In ACM Symposium on Mobile Ad Hoc Networking and Computing*, Long Beach, CA, October 2001.
9. Gergely V. et. al. "Bluetrees - Scatternet Formation to Enable Bluetooth-Based Ad Hoc Networks". *In Proceedings of IEEE International Conference on Communications*, 2001
10. Shivanajay, M., Tham, CK. and Srinivasan, D. “Mobile Agents Based Routing Protocol for Mobile Ad Hoc Networks”, *In Proceedings of IEEE GLOBECOM 2002*, November 2002, Taipei, Taiwan.
11. Minar, N. et. al. “Cooperating Mobile Agents for Dynamic Network Routing.” *Software Agents for Future Communications Systems*, Springer-Verlag, 1999.
12. Lange, D. and Oshima, M. “Seven Good Reasons for Mobile Agents.” *Communications of the ACM*, March 1999.
13. Vuong, S and Mathy, M. “Simulating The Mobile-IP Protocol Using Wave.” *EtaCOM '96 - The First International Conference on Emerging Technologies and Applications in Communications*, IEEE Computer Society, Portland, May 1996.
14. González-Valenzuela, S. and Leung, V.C.M., “QoS-Routing For MPLS Networks Employing Mobile Agents”. *IEEE-Network, Special Issue on the Applicability of Mobile Agents to Telecommunications*, May-June 2002.
15. Sapaty, P. “Mobile Processing in Distributed and Open Environments.” *John Wiley and Sons*, 2000.
16. Vuong S. and Ivanov, I. “Mobile Intelligent Agent Systems: Wave Vs. Java.” *EtaCOM '96- The First International Conference on Emerging Technologies and Applications in Communications*, IEEE Computer Society, Portland, May 1996.
17. Broch, J. et. al. “A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols.” *In Proc. of ACM/IEEE MobiCom*, October 1998.
18. Tan, G. et. al., “An Efficient Scatternet Formation Algorithm for Dynamic Environments”, *In proceedings of IASTED*, November 2002.
19. Gonzalez-Valenzuela, S and Vuong, S. T. “Evaluation of Migration Strategies for Mobile Agents in Network Routing.” *In proceedings of MATA’02*, Barcelona, Spain, October 2002
20. The Wave Page, http://www-zorn.ira.uka.de/wave/wave.html. Hosted by the Faculty of Informatics, University of Karlsruhe, Germany.

Specification and Selection of Network Management Agents

Ichiro Satoh[*]

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan

Abstract. This paper proposes a framework for building a reusable mobile agent from two kinds of components: an itinerary component and an application-specific logic component. Both components are implemented as mobile agents. The former is a carrier of the latter over particular networks and the latter defines management tasks performed at each host independently of any network. This framework also provides a mechanism for matchmaking the two mobile agent-based components. Since the mechanism is formulated based on a process algebra approach, it can strictly select an itinerary component that is suitable to perform management tasks at hosts that the tasks want to visit over networks. A prototype implementation of this framework and its application were constructed on a Java-based mobile agent system.

1 Introduction

Network management for telecommunication systems is is a typical application of mobile agent technology. Adopting mobile agent technology eliminates the need for the administrator to constantly monitor many network management activities, e.g., installation and upgrading of software and periodic auditing of the network. There have been several attempts to apply this technology to network management tasks.

There has been a serious problem associated with the development of mobile agent-based network management systems in addition to security problems. Such systems are required to efficiently migrate their agents among all specified multiple hosts because the itineraries of agents seriously affect the achievement and efficiency of network management tasks. Network management systems on the other hand must often handle incomplete networks that may have various malfunctions and disconnections and whose exact topology may not be known. It is almost impossible to dynamically generate an efficient itinerary among multiple hosts. As a result, many existing mobile agent-based network management systems explicitly and implicitly assume that their mobile agents have been statically designed for particular itineraries over their target networks. However, such an agent that has been optimized for particular networks cannot be reused in other networks.

To solve this, we constructed a framework for building and operating mobile agents for network management without losing their reusability or efficiency. The framework

[*] E-mail: ichiro@nii.ac.jp

separates the application-specific tasks and itineraries of mobile agents. The former defines network management tasks independently of any networks and the latter can be optimized for particular networks. The framework also offers a mechanism for matchmaking between the two. Since the mechanism is formulated based on an extended process algebra for reasoning about the itineraries of mobile agents, it can select an appropriate itinerary that can satisfy the requirements of a network management task. The current implementation of the framework is built on a Java-based mobile agent system, called MobileSpaces [9].

This paper is organized as follows: Section 2 presents the basic ideas behind this framework and Section 3 defines the process algebra for specifying mobile agents. Section 4 describes a prototype implementation of the framework and Section 5 presents some applications. Section 6 surveys related work and Section 7 sums up with concluding remarks.

2 Approach

The goal of this paper is to propose a framework for building and operating mobile agents, which can autonomously travel among hosts on multiple sub-networks to perform their management tasks at each of the hosts they visit.

2.1 Two-Layered Mobile Agents

The framework divides a mobile agent for network management into two layered mobile agents as follows:

Navigator Agent is independent of any application-specific tasks. Instead, it has its own itinerary on a sub-network and carries task agents among their multiple destinations on the sub-network. It is reused with arbitrary network management tasks.

Task Agent is an application-specific agent that performs its management task at each of the hosts it visits. It can travel from sub-network to sub-network, but may be unfamiliar with the sub-networks it visits. It can be reused in other sub-networks.

Most mobile agents for network management, which itinerate among multiple hosts, often perform the same code, such as monitoring and controlling various equipments, at each of the hosts they visit. Therefore, task agents do not always have to change their tasks according to its visiting hosts.

2.2 Mobile Agent Matchmaking Mechanism

This framework also provides a mechanism for matchmaking between tasks agents and navigator agents. The mechanism, called Agent Pool, stores idle agents in a manner similar to that in a bus-terminal or a taxi stand (Fig. 1). Each sub-network has more than one agent pool for storing navigator agents with various itineraries. Each task agent is responsible for traveling among the agent pools of its destination sub-networks, where each navigator agent is responsible for navigating its inner agents among the hosts in its sub-network, and has been designed to return to its place soon after achieving its

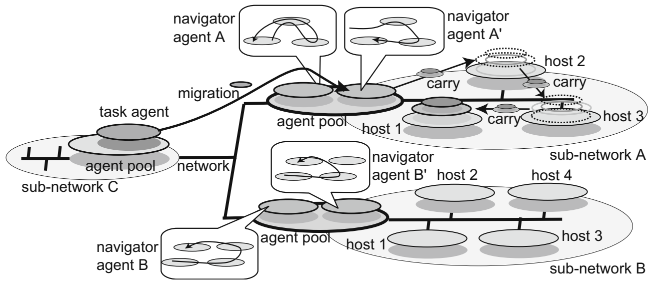


Fig. 1. Agent pools, navigator agents, and task agent

navigation task to wait for the next task. Therefore, to travel among some of the hosts on a sub-network, a task agent migrates to an agent pool at the sub-network and selects a navigator agent stored in the pool to carry it among the hosts. Also, each agent pool should assign a task agent to idle navigator agents, which are staying at agent pools, since moving agents are busy in achieving their current tasks.

Since mobile agents are written in general-purpose programming languages, such as Java, it is difficult to extract only the itineraries of mobile agents from their programs. We therefore defined a process algebra-based specification language for the itineraries of mobile agents to select mobile agents according to their itineraries. This framework assumes that each task agent specifies its required itinerary as a term of the language and each navigator agent specifies its own possible itinerary as a term of a subset of the language. The selection of navigator agents is formulated based on an algebraic order relation over the terms of the language.

3 Mobile Agent Selection

A typical mobile agent for network management must monitor and control some equipment at multiple hosts over a network whose exact topology may not be known and which may have various malfunctions and disconnections. Such an agent often has its own itinerary to statically solve problems in its target network. When a task agent is carried by a navigator agent, the performance and achievement of the task agent is dependent on the itinerary of the navigator. If a mobile agent gathers information from a host and reflects the information on other hosts, its order of movement among these hosts may affect their states. Therefore, such an agent must migrate among hosts according to a specified itinerary. However, if an agent can travel among hosts to aggregate interesting information from them without writing on them, the order of movement may be independent of its achievement. Moreover, an agent's itinerary is often dependent on the results of the agent's network management task. For example, such an agent can determine its destinations based on information, such as routing tables, it has acquired from the hosts that it has visited so far.

Definition 3.1 The set $\mathcal{E}$ of expressions of the language, ranged over by $E, E_1, E_2, \ldots$ is defined recursively by the following abstract syntax:

$$E ::= \mathsf{0} \quad | \quad \ell \quad | \quad E_1 \; ; \; E_2 \quad | \quad E_1 + E_2 \quad | \quad E_1 \# E_2 \quad | \quad E_1 \,\%\, E_2 \quad | \quad E_1 \,\&\, E_2 \quad | \quad E^{*}$$

where $\mathcal{L}$ is the set of location names, ranged over by $\ell, \ell_1, \ell_2, \ldots$. We often omit $\mathtt{0}$. We describe a subset language of $\mathcal{E}$ as $\mathcal{S}$, when eliminating $E_1 \mathbin{\#} E_2$, $E_1 \mathbin{\%} E_2$, $E_1 \mathbin{\&} E_2$, and $E^{\star}$ from $\mathcal{E}$. Let $S, S_1, S_2, \ldots$ be elements of $\mathcal{S}$. □

This framework assumes that each agent has its own itinerary written in $\mathcal{S}$. Since each agent has an interpreter for the terms of $\mathcal{S}$, it can dynamically evaluate its itinerary and migrate among hosts along the itinerary. Intuitively, the meanings of the construction are as follows: $\mathtt{0}$ represents a terminated itinerary; ℓ represents agent migration to a host whose name or network address is ℓ; $E_1 \,;\, E_2$ denotes the sequential composition of two itineraries E_1 and E_2. If the migration of E_1 terminates, then the migration of E_2 follows that of E_1; $E_1 + E_2$ denotes that an agent moves according to either E_1 or E_2 where selection can be explicitly done by processing the agent; $E_1 \mathbin{\#} E_2$ means that an agent can select either E_1 or E_2 under its control regardless of its processing; $E_1 \mathbin{\%} E_2$ means that an agent can follow either E_1 before E_2 or E_2 before E_1 as its itinerary; $E_1 \mathbin{\&} E_2$ means that two itineraries E_1 and E_2 can be performed asynchronously[1]; $E^{\star}$ is the transitive closure of E and means that an agent can move along E an arbitrary number of times. The formal semantics of the language is defined as the following labeled transition rules:

Definition 3.2 The language is a labeled transition system $\langle\, \mathcal{E},\, \mathcal{L} \cup \{\tau\}\, \{ \xrightarrow{\alpha} \subseteq \mathcal{E} \times \mathcal{E} \mid \alpha \in \mathcal{E} \cup \{\tau\} \,\}\, \rangle$ defined as induction rules as given below:

$$\frac{-}{\ell \xrightarrow{\ell} \mathtt{0}} \quad \frac{E_1 \xrightarrow{\ell} E_1'}{E_1 \,;\, E_2 \xrightarrow{\ell} E_1' \,;\, E_2} \quad \frac{E_1 \xrightarrow{\ell} E_1'}{E_1 + E_2 \xrightarrow{\ell} E_1'} \quad \frac{E_2 \xrightarrow{\ell} E_2'}{E_1 + E_2 \xrightarrow{\ell} E_2'}$$

$$\frac{E_1 \xrightarrow{\ell} E_1'}{E_1 \mathbin{\&} E_2 \xrightarrow{\ell} E_1' \mathbin{\&} E_2} \quad \frac{E_2 \xrightarrow{\ell} E_2'}{E_1 \mathbin{\&} E_2 \xrightarrow{\ell} E_1 \mathbin{\&} E_2'}$$

$$\frac{-}{E_1 \mathbin{\#} E_2 \xrightarrow{\tau} E_1} \quad \frac{-}{E_1 \mathbin{\#} E_2 \xrightarrow{\tau} E_2} \quad \frac{-}{E_1 \mathbin{\%} E_2 \xrightarrow{\tau} E_1 \,;\, E_2} \quad \frac{-}{E_1 \mathbin{\%} E_2 \xrightarrow{\tau} E_2 \,;\, E_1}$$

$$\frac{E_1 \xrightarrow{\tau} E_1'}{E_1 \,;\, E_2 \xrightarrow{\tau} E_1' \,;\, E_2} \quad \frac{E_1 \xrightarrow{\tau} E_1'}{E_1 + E_2 \xrightarrow{\tau} E_1'} \quad \frac{E_2 \xrightarrow{\tau} E_2'}{E_1 + E_2 \xrightarrow{\tau} E_2'}$$

$$\frac{E_1 \xrightarrow{\tau} E_1'}{E_1 \mathbin{\&} E_2 \xrightarrow{\tau} E_1' \mathbin{\&} E_2} \quad \frac{E_2 \xrightarrow{\tau} E_2'}{E_1 \mathbin{\&} E_2 \xrightarrow{\tau} E_1 \mathbin{\&} E_2'}$$

where $\mathtt{0} \,;\, E$ is treated to be syntactically equal to E and $E^{\star}$ is recursively defined as $\mathtt{0} \mathbin{\#} (E \,;\, E^{\star})$. We often abbreviate $E_0 \xrightarrow{\tau} E_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} E_{n-1} \xrightarrow{\tau} E_n$ to $E_0 (\xrightarrow{\tau})^n E_n$. □

In Definition 3.2, the ℓ-transition defines the semantics of an agent's mobility. For example $E \xrightarrow{\ell} E'$ means that the agent moves to a host named ℓ and then behaves as E'. Also, if there are two possible transitions $E \xrightarrow{\ell_1} E_1$ and $E \xrightarrow{\ell_2} E_2$ in an agent, the agent being processed chooses one of destinations ℓ_1 and ℓ_2. However, the τ-transition corresponds to a non-deterministic choice in an agent's itinerary. Let us describe three agent migration patterns studied in [1]. To simplify our discussion hereafter, we introduce three macros, corresponding to the patterns, e.g., *Travel*, *Star*, and *Turn*. These

[1] In CCS-like process algebras, $\mathbin{\&}$ is an operator for specifying parallel executions. The operational semantics of the language is an interleaving model in the literature of process algebras, and each agent migration is an atomic action.

macros do not extend the language because they are mapped into $\mathcal{E}$. We will describe the list of host names as $[\ell_1, \ell_2, \ldots, \ell_n]$, where $\ell_1, \ldots, \ell_n \in \mathcal{L}$. Let [] be an empty list, $car(X)$ be the top element of list X, i.e., let ℓ_1 and $cdr(X)$ be the remaining list of X except for the top element, i.e., $[\ell_2, \ldots, \ell_n]$.

$$\begin{aligned}
Travel(\$(\mathrm{X})) &\stackrel{\mathrm{def}}{=} car(\$(\mathrm{X}))\ ;\ Travel(cdr(\$(\mathrm{X})))\\
Travel([]) &\stackrel{\mathrm{def}}{=} 0\\
Star(\$(\mathrm{X})|h) &\stackrel{\mathrm{def}}{=} (car(\$(\mathrm{X}))\ ;\ h)\ ;\ Star(cdr(\$(\mathrm{X}))|h)\\
Star([]|h) &\stackrel{\mathrm{def}}{=} 0
\end{aligned}$$

Let h be an element of $\mathcal{L}$ and X be a list of host names in $\mathcal{L}$. To illustrate the transition defined in Definition 3.2, we shows a partial transition of $Star([a, b, c]|h)$ as follows:

$$\begin{aligned}
Star([a,b,c]|h) &\stackrel{\mathrm{def}}{=} (a\ ;\ h)\ ;\ Star([b,c]|h)\\
&\xrightarrow{a} h\ ;\ Star([b,c]|h)\\
&\xrightarrow{h} Star([b,c]|h)\\
&\stackrel{\mathrm{def}}{=} (b\ ;\ h)\ ;\ Star([c]|h)\\
&\xrightarrow{b} h\ ;\ Star([c]|h)\\
&\xrightarrow{h} Star([c]|h)
\end{aligned}$$

We next formulate an algebraic order relation based on the concept of bisimulation [7]. The relation is suitable for selecting one of the navigator agents whose itineraries can satisfy the requirement of a task agent.

Definition 3.3 A binary relation $\mathcal{R}^n$ ($\mathcal{R} \subseteq (\mathcal{E} \times \mathcal{S}) \times \mathcal{N}$) is an *n-itinerary* prebisimulation, where $\mathcal{N}$ is the set of natural numbers. If whenever $(E, S) \in \mathcal{R}^n$ where $n \geq 0$, then the following hold for all $\ell \in \mathcal{L}$ or τ.

(i) if $E \xrightarrow{\ell} E'$ then there is an S' such that $S \xrightarrow{\ell} S'$ and $(E', S') \in \mathcal{R}^{n-1}$
(ii) $E\,(\xrightarrow{\tau})^*\,E'$ and $(E', S) \in \mathcal{R}^n$
(iii) if $S \xrightarrow{\ell} S'$ then there exist E', E'' such that $E\,(\xrightarrow{\tau})^* E' \xrightarrow{\ell} E''$ and $(E', S') \in \mathcal{R}^{n-1}$

where $E \sqsupseteq_n S$ if there exist some n-itinerary prebisimulations such that $(E, S) \in \mathcal{R}^n$. We call $\sqsupseteq_n$ *n-itinerary* order. □

The informal meaning of $E \sqsupseteq_n S$ is that S is included in one of the permissible itineraries specified in E and n corresponds to the number of movements of the agent that can satisfy E. There are some basic examples below.

- $(a \,\%\, b \,\%\, c)\ ;\ h \sqsupseteq_4 c\ ;\ a\ ;\ b\ ;\ h$
  where the right side requires an agent to migrate among three hosts a, b, and c in indefinite order and then return to host h and the right side migrates among three hosts c, a, and b sequentially. When the left side is changed to $a\ ;\ b\ ;\ c\ ;\ h$, the relation is still preserved, but when the left side becomes $a\ ;\ h\ ;\ b\ ;\ h\ ;\ c\ ;\ h$ or $a\ ;\ b\ ;\ h$, the relation is not preserved.
- $((a\ ;\ b\ ;\ c)\ \&\ h^*)\ ;\ h \sqsupseteq_6 a\ ;\ h\ ;\ b\ ;\ h\ ;\ c\ ;\ h$
  where the left side allows an agent to drop in at host h at arbitrary times during the itinerary $a\ ;\ b\ ;\ c$ and then finish its movement at host h. The right is a star-shaped route between three destinations, a, b, c and host h can satisfy the left side.

4 Design and Implementation

This section presents a prototype implementation of our framework. We tried to keep the implementation within the framework as much as possible.

4.1 Hierarchical Mobile Agents

Before describing the framework presented in this paper, let us briefly review the MobileSpaces mobile agent system that has provided the infrastructure for this framework. [2] Mobile agents in MobileSpaces are programmable entities like other mobile agents. They are capable of conserving their state while on the move and their itineraries can include multiple hosts. Furthermore, MobileSpaces provides each mobile agent with two novel concepts: *agent hierarchy* and *group migration*. The former means that another mobile agent can be contained within a mobile agent. The latter means that each mobile agent can migrate to another mobile agent or computer along with all its inner agents, as long as the destination accepts them. Therefore, an agent can contain other mobile agents inside it. Each agent has direct control of all its inner agents and can thus instruct them to move to other locations and destroy them. In contrast, each agent has no direct control over its container agents. Instead, each agent can have a set of service methods, which can be accessed by its containers. Each agent has a globally unique name and can have more than one active thread under the control of the runtime system.

4.2 Navigator Agent

Each navigator agent is a container of one or more task agents and is responsible for carrying them to hosts in the network it covers. It travels with its inner agents in accordance with its itinerary written in $\mathcal{S}$ and invokes the callback methods of its inner task agents at certain times, such as arrival and departure. Each navigator agent is designed to go back to its agent pool and then register its itinerary at the pool soon after achieving its navigation to wait for the next task. This framework provides abstract classes in the Java language and navigator agents can be defined by extending these classes.

```
public class NavigatorAgent extends MobileAgent {
  void setRoute(Route r) throws ... { ... }
  void moveTo(Host h) throws IllegalHostException,
    NoSuchHostException ... { ... }
  void moveToNext() throws MultiplePossibleHostsException,
    NoSuchHostException ... { ... }
  Host[] getPossibleHosts() ... { ... }
  void arrivedAt(Host here);
  void depaturingFor(Host dst);
  ...
}
```

Each navigator agent has its own itinerary as a term of $\mathcal{S}$ and registers the term with itself and an agent pool, in which it is stored, by invoking the `setRoute()` method as follows:

[2] Details on the MobileSpaces mobile agent system can be found in our previous paper [9].

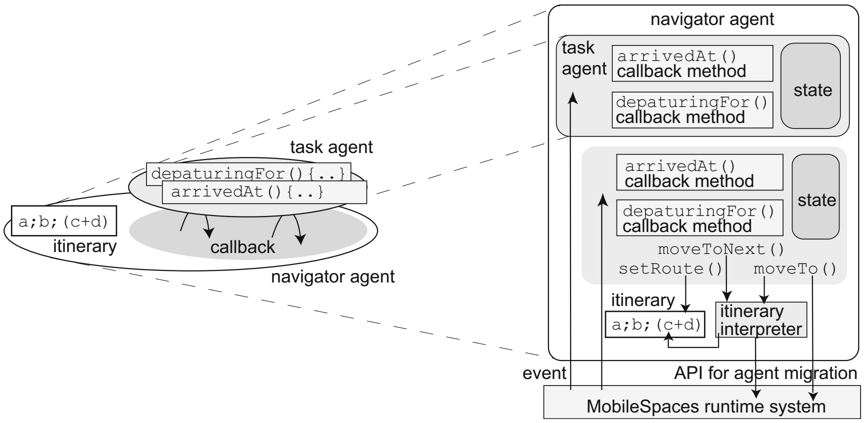


Fig. 2. Structure of navigator agent

```
setRoute(new Route("a;b;(c+d)"));
```

where `a;b;(c+d)` is an itinerary attached to the navigator agent and means that the agent migrates to host `a` and then to host `b`. The agent can then select either host `c` or `d` according to its own processing results. Each agent can migrate itself over a network by using the following two approaches.

The first approach allows each agent to move along the itinerary it has registered with itself. Each agent has a lightweight interpreter for the language in $\mathcal{S}$. When the agent invokes the `moveToNext()` method, the interpreter evaluates the agent's next destination from the itinerary and automatically moves the agent to the destination. However, if the itinerary contains one or more candidate destinations combined by selective operator `+`, then the invocation of the method throws an exception, named `MultiplePossibleHostsException`. The agent can get all the destinations that it can move to at the next hop by invoking the `getPossibleHosts()` method and it moves to one of these by invoking the `moveTo(dst)` method with the selected destination specified as `dst`. For example, suppose that an agent registers `a;b;(c+d)` as its own itinerary. As we can see Fig. 3, it performs method `moveToNext()` two times for two hops; from the current host to `a` and then from host `a` to `b`. It can then select either `c` or `d` and then perform the `moveTo(dst)` method with the name of the selected destination as the method's argument.

The second approach corresponds to the common approach used in existing mobile agent systems. That is, an agent explicitly specifies its destination whenever it migrates itself over a network. The `moveTo()` of the `NavigatorAgent` class causes the agent to move from host `b` to the destination specified as its argument. For example, an agent whose itinerary is `a;b;(c+d)` can invoke the `moveTo()` method with `a` and then `b` to move to host `a` and then to `b`. It can then invoke the same method with either `c` or `d`.

This framework restricts navigator agents from straying from the itinerary they registered with themselves. In both the above approaches, when the movement of a mobile agent deviates from the itinerary registered by invoking the `setRoute()` method, it is constrained and `IllegalHostException` is thrown to the agent. Each navigator agent can explicitly limit the length of the execution period for its incoming task agents after arriving at each destination. When the time limit of a task agent inside it expires, it automatically terminates the task agent. Each agent can dynamically register its itinerary by invoking the `setRoute()` method while it is moving, but the new itinerary only becomes available after it returns to a certain agent pool.

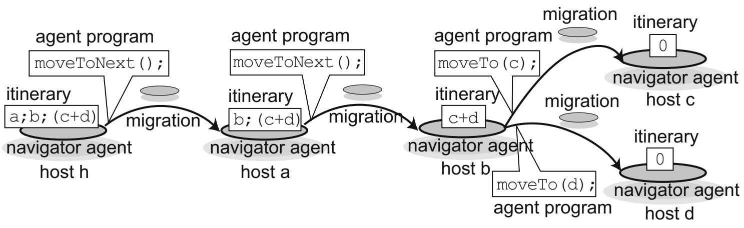


Fig. 3. Following-itinerary movement of a mobile agent whose itinerary is specified as `a;b;(c+d)`.

4.3 Task Agent

Each task agent is a mobile agent that defines its management tasks at each of the hosts in accordance with its management criteria. Although it may be able to travel among the agent pools of its target sub-networks, it is unfamiliar within each of the sub-networks. This framework provides a Java-based abstract class that allows us to easily define advanced task agents by extending the the `TaskAgent` class.

```
public class TaskAgent extends MobileAgent {
  void setRoute(Route r)
    throws IllegalSyntaxException ... { ... }
  void arrivedAt(Host here);
  void depaturingFor(Host dst);
  void finished(Route r);
  ...
}
```

Each agent defines its task in the `arrivedAt()` method. When arriving at an agent pool, a task agent gives the pool the required itinerary along which a navigator agent needs to carry itself by performing the `setRoute()` method with a term of $\mathcal{E}$ corresponding to that itinerary. The agent pool selects a suitable navigator agent and then migrates the task agent into the selected agent. Having arrived at a host, the navigator agent invokes the `arrivedAt()` method of its task agent to instruct it to do something during a given time period at the host. After receiving a certain event from all the task agents or after the period has elapsed, the navigator agent invokes the `depaturingFor()` method with the address of the next host and then moves itself and its task agents to the destination according to its itinerary. For reasons of security, all agents must be authenticated by the agent pool of a sub-network on behalf of the sub-network. This is helpful in network management systems whose hosts may have limited CPU power and memory. Since a sub-network may explicitly prohibit any task agent from visiting its hosts, task agents must be carried by a navigator agent managed by the agent pool of the sub-network. Therefore, a task agent alone cannot migrate to all the hosts, even if it knows the addresses of its target hosts in the sub-network.

4.4 Agent Pool

Each agent pool is a stationary agent, which can contain more than one navigator agent as shown in Fig. 4. It is also responsible for receiving the requirements of a visiting task agent and selecting a suitable navigator agent to carry the task agent among hosts on its sub-network. It maintains a repository database about the itineraries of idle navigator agents waiting for a chance to guide task agents. When an agent pool receives a task agent, it extracts the required itinerary from it and selects a navigator agent whose itinerary can satisfy the required itinerary from the idle navigator agents stored inside it. The selection mechanism is just a direct implementation of the algebraic relation presented in Definition 3.3. That is, the agent pool compares the required itinerary written in $\mathcal{E}$ with each of the possible itineraries of the agents written in $\mathcal{S}$ by directly using the order relation $\sqsupseteq_n \subseteq \mathcal{E} \times \mathcal{S}$ in Definition 3.3. It then assigns the task agent to the navigator agent whose itinerary can satisfy the itinerary required. If more than one navigator agent satisfies the required itinerary, it selects an agent with the least number of agent migrations over a network, which is n of $\sqsupseteq_n$ in Definition 3.3.

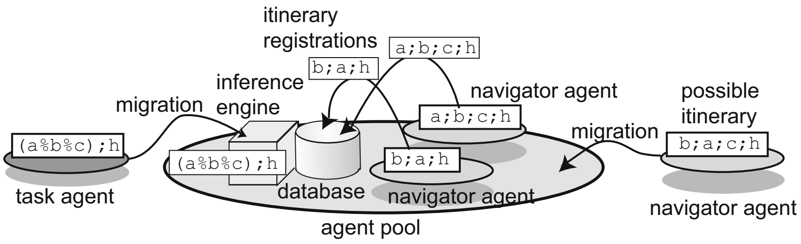


Fig. 4. Agent pool

4.5 Current Status

The cost of selecting navigator agents is dependent on the number of agents and the length of itineraries. Although the current implementation was not optimized for performance, it can handle each of all the itineraries presented in this paper within a few milliseconds. Also, the per-hop latency of migrating a simple task agent using a navigator agent whose itinerary is static is 43 ms. This is where the per-hop latency of agent migration using a non-hierarchical minimal-size agent is 35 ms in a MobileSpaces runtime system running on computers (Pentium III-800 MHz with Windows2000 and JDK 1.4) connected through a 100-Mbps Ethernet.

5 Application

We developed a network management system for a cluster computing environment consisting of two sub-networks to evaluate the effectiveness of this framework. Each of the sub-networks had from four to eight processor elements distributed geographically. [3]

[3] We presented the GRID environment in our previous paper [11] It was small scale because it was implemented as a testbed for developing middleware and applications for GRID computing rather than a computational infrastructure.

The purpose of this system was to monitor various network and computational resources at the hosts.

The system deploys an agent pool at one host of each sub-network and offers several task agents and navigator agents. For example, the task agent that monitors network traffic loads has been designed to perform its task at each host that it visits. Although the system itself is independent of any network management protocols, we constructed a task agent that can access SNMP data from a small stationary agent situated at its visiting host. The stationary agent allows that visiting task agent to access the MIB of its host via interagent communication. Since the task agent can contain codes to perform both information retrieval and filtering, it can only carry relevant information. Also, the system has three other task agents for monitoring computational resources at the processor hosts. They have been designed to collect information on the use of CPU, memory, and disks by incorporating performance monitoring systems at the hosts. Also, the system offers several navigator agents with different itineraries. However, due to word limitations, this section only explains two navigator agents optimized for one of the sub-networks defined by `NaviAgent1` and `NaviAgent2` classes.

```
 1:  public class NaviAgent1 extends NavigatorAgent {
 2:    public NaviAgent1() {
 3:      // registering its possible itinerary
 4:      setRoute(new Route("h;a;b;c;d;h"));
 5:    }
 6:    // invoked at the completion of the task agent's
 7:    // processing at the current host
 8:    public void done() throws
 9:      MultiplePossibleHostsException .. {
10:      moveToNext();
11:    }
12:    ...
13:  }
```

`NaviAgent1` can travel along a sequential route. `NaviAgent2` is defined as the same class except for the fourth line as follows:

```
 4:      setRoute(new Route("h;a;h;b;h;c;h;d;h"));
```

The above itinerary defines a star-shaped route among four hosts,`a`, `b`, `c`, and `d`. Next, suppose a task agent gathers local information from SNMP agent running on each of the hosts that it visits. The agent has its required itinerary specified as follows:

```
 1:  public class SimpleTaskAgent extends TaskAgent {
 2:    public SimpleTaskAgent() {
 3:      setRoute(new Route("h;([SNMP-AGENT]&h^*);h"));
 4:    ...
 5:    }
 6:    ...
 7:  }
```

where `[SNMP-AGENT]` is a list of the hosts that perform SNMP agents. We assumed that the list could be transformed into an itinerary (`a % b % c % d`), which means that the agent must visit the four hosts specified in `a`, `b`, `c`, and `d` in any order of movement and can visit the home host `h` more than zero times on the way. The two navigator agents can satisfy the required itinerary of the task agent. Since `NaviAgent1` has fewer agent migrations than that for `NaviAgent2`, the agent pool selects the former navigator agent and moves the task agent into it. After receiving the task agent, the `NaviAgent1` navigator agent carries it from host to host according to its own itinerary. Whenever it arrives at one of the destinations, it issues certain events to invoke the `arrived()` method of the task. The task agent performs its application-specific task, such as accessing and filtering from the SNMP agent of its visiting host, defined in the `arrived()` method.

6 Related Work

Many mobile agent systems have been developed over the last few years. There have been several attempts to develop mobile agent-based network management, for example see [2, 4]. Existing work on mobile agents has focused on the development of agent infrastructures, applications, and functions that can be used by agents, but not on approaches to selecting mobile agents. Of these, Plangent [8] is a mobile agent system, which can dynamically generate a plan to let itself acquire the knowledge that users need and then perform its application-specific actions and movements according to the plan. However, Plangent's planning functionality does not target the mobility of agents and cannot always generate valid plans. Our approach on the other hand offers a theoretical foundation for the selection of mobile agents and allows us to determine whether or not the movements of agents can satisfy the requirements of network management tasks.

Moreover, there have been various theoretical models for specifying mobile agents, e.g., Mobile UNITY [6] and Ambient calculus [3]. The former is an extension of UNITY and was designed for specifying control flows, variables, and conditional assignment statements at programs; it cannot merely extract or analyze only the itineraries of mobile components. The latter is just a theoretical framework for formalizing the whole computation of mobile agents. As far as the author knows, no existing calculi can provide any preorder relations for the selection of mobile agents according to their itineraries.

We should next compare the framework with our previous works. We presented an approach to building mobile agents for network management in our previous paper [11]. This approach separated a mobile agent into two parts: its mobility control part and its application-specific part, like the framework presented in this paper. However, the previous paper did not provide any approaches to matchmaking two parts, unlike this paper. We presented active network protocols for building and managing several agent migration protocols in our another previous paper [10], but this was just an infrastructure for configurable protocols for agent migration.

7 Conclusion

This paper presented a framework for building and operating mobile agents for network management. The framework makes two contributions to mobile and multi-agent technologies. The first is to propose an approach for building a reusable mobile agent from two subcomponents: a navigator agent and a task agent. The second contribution is to formulate a specification language and an algebraic order relation between the terms of the language as a theoretical foundation for the selection of mobile agents. It provides a matchmaking mechanism for navigator and task subcomponents. We believe that the framework itself is general-purpose so that it is available for other mobile agent-based applications as well as network management.

Finally, we would like to mention some future research directions. We plan to establish an axiomatic system based on the order relation to improve the performance of agent selection. This paper does not discuss any coordination among multiple mobile agents, but we are interested at developing a mechanism for assigning a task to one or more mobile agents. We are interesting in applying the framework to an infrastructure for ambient intelligence in ubiquitous computing environments presented in our previous paper [12] and to mobile agent-based managements for sensor networks presented in another previous paper [13].

References

1. Y. Aridor, and D.B. Lange, Agent Design Patterns: Elements of Agent Application Design, Proceedings of Second International Conference on Autonomous Agents (Agents'98), pp. 108-115, ACM Press, 1998.
2. A. Bieszczad, B. Pagurek, and T. White: Mobile Agents for Network Management, IEEE Communications Surveys, vol. 1, no. 1, 1998.
3. L. Cardelli and A. D. Gordon: Mobile Ambients, Proceedings of Foundations of Software Science and Computational Structures, LNCS, vol. 1378, pp. 140–155, 1998.
4. A. Karmouch, Mobile Software Agents for Telecommunications, IEEE Communication Magazine, vol. 36 no. 7, 1998.
5. B. D. Lange and M. Oshima: Programming and Deploying Java Mobile Agents with Aglets, Addison-Wesley, 1998.
6. P.J. McCann, and G.-C. Roman, Compositional Programming Abstractions for Mobile Computing, IEEE Transaction on Software Engineering, vol. 24, no.2, 1998.
7. R. Milner, Communication and Concurrency, Prentice Hall, 1989.
8. A. Ohsuga, Y. Nagai, Y. Irie, M. Hattori, and S. Honiden, PLANGENT: An Approach to Making Mobile Agents Intelligent, IEEE Internet Computing, vol.1, no.4, pp.55-57, 1997.
9. I. Satoh, MobileSpaces: A Framework for Building Adaptive Distributed Applications Using a Hierarchical Mobile Agent System, Proceedings of International Conference on Distributed Computing Systems (ICDCS'2000), pp.161-168, IEEE Computer Society, April, 2000.
10. I. Satoh, Network Processing of Mobile Agents, by Mobile Agents, for Mobile Agents, Proceedings of Workshop on Mobile Agents for Telecommunication Applications (MATA'2001), LNCS, vol.2146, pp.81-92, Springer, 2001.
11. I. Satoh, A Framework for Building Reusable Mobile Agents for Network Management, Proceedings of Network Operations and Managements Symposium (NOMS'2002), pp.51-64, IEEE Communication Society, April, 2002. (A long version will appear in IEEE Transaction on Systems, Man and Cybernetics, vol.33, no.3, 2003)
12. I. Satoh, Physical Mobility and Logical Mobility in Ubiquitous Computing Environments, Proceedings of Conference on Mobile Agents (MA'02), LNCS, Vol. 2535, pp.186-202, Springer, 2002.
13. T. Umezawa, I. Satoh, Y. Anzai, A Mobile Agent-based Framework for Configurable Sensor Networks, Proceedings of International Workshop on Mobile Agents for Telecommunication Applications (MATA'2002), LNCS, Vol. 2521, pp.128-140, Springer, 2002.

Mobile Agents for Dynamic SLA Negotiation

Gilles Klein and Francine Krief

LIPN Laboratory, University of Paris XIII
Avenue Jean-Baptiste Clément 99, 93430 Villetaneuse, France.
Tel: +33 (0)1 49 40 36 12, Fax: +33 (0)148 26 07 12
{gk,krief}@lipn.univ-paris13.fr

Abstract. A Service Level Agreement (SLA) is a contract signed between a Customer and an Internet Service Provider which specifies a level of service associated to a data flow. In this paper we investigate the use of mobile agents for the dynamic negotiation of SLA between a customer and several Internet Service Providers. The proposed framework is based on Cadenus architecture and the user interface developed in the context of the RNRT project Arcade.

Keywords: Mobile agents, SLA, QoS, policy-based Management, COPS-SLS

1 Introduction

To obtain QoS guarantees in Internet, a customer signs a Service Level Agreement (SLA) with an Internet Service Provider. The SLA specifies the levels of availability, serviceability, performance, operation or other attributes of the service [1]. It contains a technical part called Service Level Specifications (SLS). SLS is a set of parameters and values which defines the service provided to a data flow. It contains especially the applicability of the SLS, the flow identifier of a SLS, the traffics conformity and performance guarantees, i.e. delay, packet loss, jitter and throughput [2].

The IETF proposes the policy-based approach as a solution to manage IP networks with QoS guarantees. Policy based management (PBM) assures dynamic configuration of network elements according to the subscribed SLA. Moreover, customer can dynamically negotiate the SLS parameters using COPS-SLS.

In the context of the RNRT project Arcade, we have proposed a user interface installed in the user terminal to determine and negotiate the SLS on behalf the user using COPS-SLS.

In this paper we propose to extend the user interface functionality to the dynamic SLA negotiation between a customer and several Internet Service Providers.

The paper is structured as follows: section 2 describes the background concepts for the purpose of this work. Section 3 presents the objectives of this work. The forth section presents the proposed framework for the dynamic SLA negotiation using mobile agents. Section 5 threats of security aspects which is a sensible point in the domain of mobile agents. The last section concludes the paper and presents our future work.

2 Background

The framework that we propose for the dynamic negotiation of the SLA is based on the CADENUS framework, agent technology and the NIA Interface. These essential components of the proposed framework are presented in the following.

2.1 The CADENUS Framework

The CADENUS project [3] proposes an integrated solution for the dynamic creation, configuration and provisioning of end-user services with QoS guarantees in premium IP network.

The CADENUS framework introduces three components to supervise the dynamic creation and service configuration process: the Access Mediator (AM), the Service Mediator (SM), the Resource Mediator (RM). The Access Mediator is independent of the underlying network through which the services are delivered. It usually deals with several service providers. It has the function of access mediation, which is to set the function of accessing to a particular service provider. It has the knowledge of the access link of the end-user and the terminal type. While optional, (the user could communicate directly with one or more service providers) the Access Mediator presents to the user a wider selection of services, ensuring the lowest cost and offering a harmonized interface. It can therefore select the cheapest offer if the same service is available from more than one provider and it can immediately notify the user that a new service becomes available.

The Service Mediator has to inform the Access Mediators of all new service offers, so that they can present these new offers to their users. It supervises the management of the physical access to the services via the appropriate underlying network, using the Resource Mediator(s). The Service Mediator maintains no direct contract with the end-users for SLAs, or for authentication or accounting. It possibly deals with other service providers to compose its services and with network providers to support its services.

The Resource Mediator is associated with the underlying network. It is responsible for the network performance demanded by the service providers. The project proposes the policy-based approach as a solution to ensure the correct operation of the network [4]. The necessary functional entities of the policy-based management framework are the Policy Manager, the Policy Enforcement Point (PEP), the policy Decision Point (PDP) and the Policy Repository.

The Policy Manager is a tool where the policy is composed and edited. It can also manage the deployment of the policy in the network.

The Policy Repository is a centralized database where all policies are stored in a structured way. The Policy Decision Point (PDP) is a logical entity responsible for retrieving and interpreting policies from Policy Repository. It has to determine which configurations need to be applied to the network resources to satisfy the policies. The PDP can work top-down in that it receives command from Policy Manager to set certain policies in the PEP (provisioning) or bottom-up in that it receives policy request from PEP (outsourcing), retrieves the requested policy and then judge if such a policy can be granted to the PEP. The Policy Enforcement Point (PEP) is a logical entity responsible for executing the actions according to the policies set by the PDP. It has to maintain local policies integrity while monitoring their enforcement.

In this approach the Resource Mediator can be seen as a PDP and network elements as PEPs.

2.2 Agent Technology

The introduction of intelligent agents in network environments is an important research topic. Various telecommunication domains are covered: web and e-business, telecommunication services and networks, software engineering [5]. This trend is motivated by the desire to use the agents to help the user (i.e. assistant agent), to perform complicated tasks (i.e. resources allocation), to automate some processes (i.e. control). An agent is a "self-contained program capable of controlling its own decision-making and acting, based on its perception of its environment, in pursuit of one or more objectives" [6]. The key properties of an agent are:

- Autonomy: the ability to function independent of the interaction of another entity (human or agent);
- Social ability: the ability to interact with others agents (software or human) in order to accomplish certain tasks or to help these agents to accomplish their tasks;
- Responsiveness: the ability to perceive the environment and respond in a required time to events occurring in it;
- Proactive ability: the ability to take the initiative whenever the situation demands;

A mobile agent incorporates these general agent properties and includes the ability to move from a network node to another.

- Cooperation: the ability to act in team in order to achieve a complex task Mobile agents move to a machine running a mobile agent server. This server provides interface with the underlying host machine and facilities to transmit, receive and communicate with these agents. Several mobile agent platforms currently are proposed, e.g. Voyager [7] and Aglets [8]. The core of an agent server is a virtual machine, onto which mobile agents are loaded and executed. Concerning the design of mobile agents, different approaches are possible [9]:
- Mobile agents transport all code and data; this design is the simplest but offers the less security.
- Mobile agents are specialized and several services are available on different servers; this design increases the complexity of the total system.

2.3 The NIA Interface

In the context of RNRT project Arcade [10], we have proposed and implemented an intelligent user interface which is called NIA (Network Interface Agent) [11]. The NIA is a multi-agent system which is installed on the user terminal and which is used for the dynamic negotiation of SLS. The SLA is previously established with the Internet Service Provider and specifies that the SLS parameters are dynamically negotiate.

The NIA determines the SLS on behalf the user according to the application requirements and the user's needs. It is composed of five agents:

– User agent: This agent identifies the user (User's identification module), continuously updates its base of knowledge of the user's needs (Data modification and memorization module) and associates the user with a profile (Behavior analysis module).
– Interface agent: This agent plays the role of intermediary between the user and the system. It contains the Message translation and Display modules.
– Application agent: It determines the type and the profile of the active application (Classification and Profile determination modules) according to the application requirements the user's needs.
– Parameters agent: This agent identifies and determines the values the SLS parameters required for the negotiation (SLS parameters and Predefined values modules). It also contains a learning module to be able to associate the best values of those parameters with the general profile of negotiation determined by the system (Values determination module).
– Control agent: It communicates the values of SLS parameters to the Service Provider (communication module) and manage renegotiations (Negotiation module). It also contains a Evaluation module. The protocol used for the SLS negotiation between the user terminal and the service provider is COPS-SLS [12].

3 Objective of This Work

The possible existence of several Access Mediators could present no further problem, if each Internet user was stuck with one of them. But we consider that it is neither probable nor desirable. One of the greatest advantages of architectures like the one described before is that it gives great freedom to the users. Being able to profit from the concurrence for every application separately may be the greatest of these liberties. To open the system so that a customer could choose the Access Mediator that offers him the best services at the lowest price, we conceived an architecture based on software agents that would survey and negotiate the offers of the different Access Mediator for the user.

According to the profile of the user, the NIA interface introduced in the user's terminal could communicate the user preferences and the SLS parameters to mobile agents and select the service provider who best answers the application requirements and user's needs. It could change the service provider with the modification of the user's profile or with the presentation of more interesting offers.

This framework presents many advantages: the function of the Access mediator is confirmed and his utility increases, an ISP can more easily propose new services by using it, the role of the NIA interface is extended to the choose of the best service provider available at certain moment.

4 Proposed Framework

This extension of the NIA interface uses three different types of agents, Users Negotiators (UN) which are mobile agents sent by the customer to the Access Mediator platform to negotiate services and survey the offers and rates of the Access Mediator, Access Negotiators (AN) which negotiate services and rates in favour of the Access Mediators and User Overseer (UO) which manages the information sent by the ANs and makes the final decisions.

The simplest solution would have been to consider that the Access Mediators create a great market place where they could all propose their offers and negotiate with potential customers. But this is not a realistic solution as the Access Mediators are mostly competitors which have no reasons to co-operate.

So we proposed another solution: every Access Mediator could open access to a multiagent platform on its site that would welcome UN agents from customers. These agents would be authenticated (for example by the banks of their owners) and then computer power would be allocated to them depending on the services they subscribed to (and paid). These agents would then survey the offers made by the Mediator Access, filter them and send the most interesting ones to their different owners' Overseers (see figure 1). Here, the autonomy which defines agents is necessary as the UNs are not only an independent negotiator but also agents filtering information and sending only the most pertinent pieces of knowledge to the UO. So the UO has to give them the user's and application's profiles then they have to take decision based on them. Finally agent technologies offer the use of negotiation protocols which are the only acceptable method for reaching an agreement between the different parties (the Access Mediator and the user).

When a computer user needs a specific service, his overseer reviews the different offers and price-lists sent by the UN, selects several Access Mediators whose offers are interesting and sends his needs to the UN based on these Mediators' platforms. Then, the UNs open negotiations with their AN counterparts to obtain the best rates and quality of services. When the UN and the AN reach an agreement (or after a certain length of time, if no agreement is reached), the UN sends the description of the best proposal made by the AN to the Overseer. Then, the Overseer compares the different offers it received from the different ANs and sends back his agreement to the best one. Then every agent concerned by the different concurrent negotiations returns to its normal state. Figure 2 describes the interactions between agents using AUML.

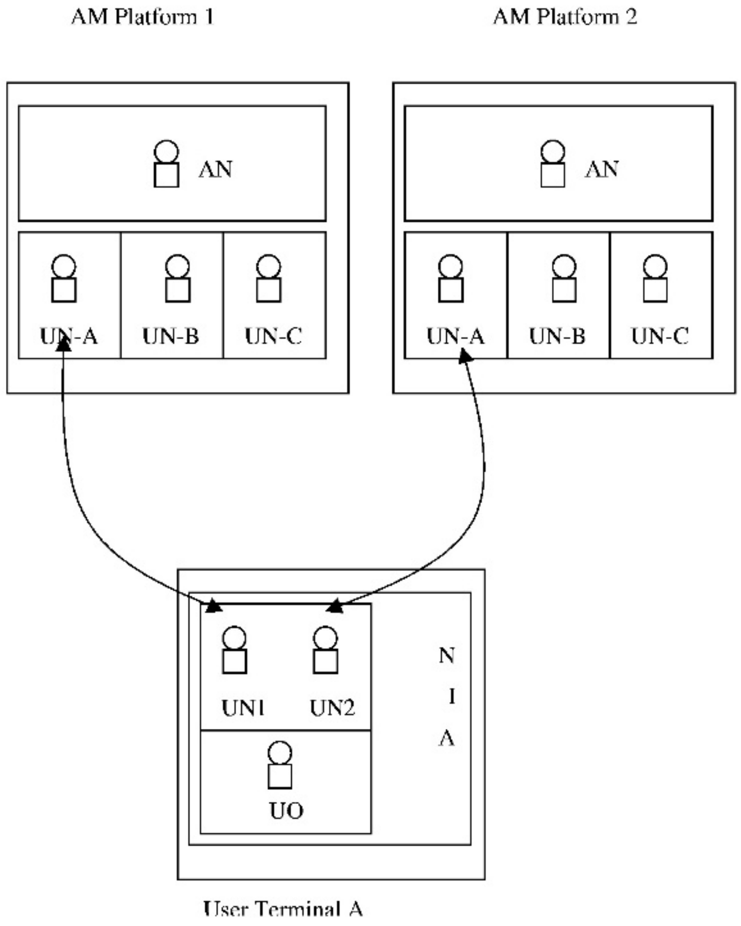


Fig. 1. Framework for dynamic SLA negotiation

5 Security Aspects

Implementing this framework presents several difficulties that must not be overlooked:

The agents must be authenticated before it can access the platform. Authentication is a common problem of electronic business, but it is stronger here, as the customer is not only buying a good, he is renting a service that can be used to commit villainous acts for which the provider could be hold responsible. So the Access Mediator must be guarantied that the authentication is foolproof. The best way to obtain that is through an accepted authority. The simplest solution is through the agent owner's bank.

The second sensible point is the honesty of the Access Mediator. As it provides the platform the agents migrate to, it can access every piece of information

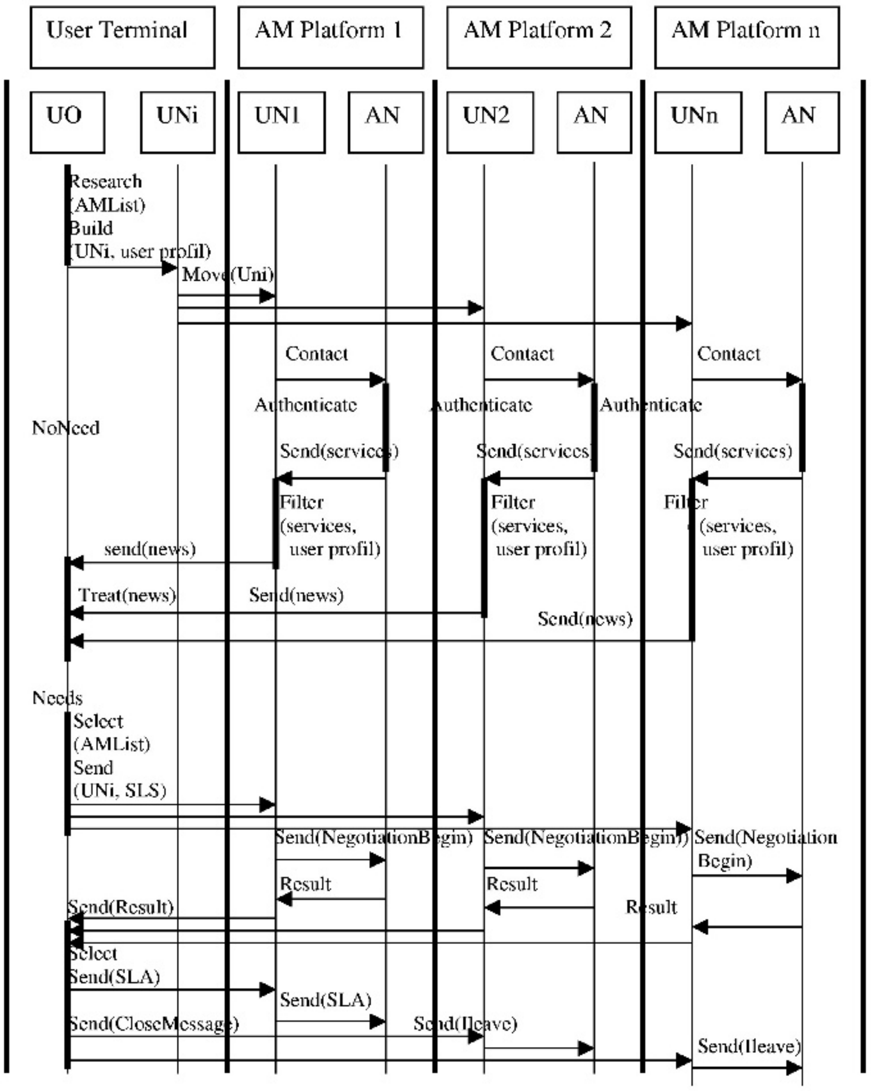


Fig. 2. AUML representation of interactions between agents

included in the User Negotiator agent and monitor every communication from and to the agent. It can be a problem, as it can give him a serious edge during the negotiations. In fact, it even could falsify messages to the overseer. It can be a serious breech, but as long as the information transmitted between the different agents of our system are only about the needs of the user, it is not really important as these pieces of information are finally meant to be fulfilled by the AN and should not contain anything really private. The second part is that there are several UNs negotiating concurrently with several competing ANs so as long as the different Access Mediators are not plotting together, short of lying about the services sold, there is no point cheating.

The third sensible point is the payment. We must guaranty that the customer pays for the services he got (nothing more, nothing less). To obtain this result, the simplest solution is to implement continuous electronic payment. To work correctly, usurping the identity of another user must be very difficult, that is why the best solution would be to use the user's bank a s a naming authority.

6 Conclusion and Future Work

Permitting the existence of a real competition between the different services providers would be a serious breakthrough for the Internet consumers. It does not solve every problem as certain geographical areas are connected only through a single Internet Service Provider, so that service can never really be subject to competition. Therefore, the important development of IP mobile terminals and wireless networks make a framework for dynamic SLA negotiation more attractive. Our system could also permit users to buy a minimum Internet subscription and then contract with the cheapest access mediator the lowest price for specific services when the needs arise. However, the providers (access mediators) can also gain from this system as they can rent negotiator agents power that may even not be used most of the time. They also can more easily notify the user that a new service becomes available.

Introduction of reinforcement learning could be interesting to privilege the offers' ISP providing the best satisfaction for the user. To make this system viable, it will be necessary to calculate the cost for the access mediators in terms of computer might (and also in terms of communications). To work correctly, this model needs the agents to be able to take options on certain services, at least long enough for the Overseer agent to decide which provider will finally be chosen. These options are the greatest single difference between the original arcade model that we did not analyse fully yet. It would also be good to be able to practise a real scale test.

However, competing services will appear rapidly and every service provider should ready itself to confront the fact.

References

[1] A. Westerinen et al., “Terminology for Policy-Based Management”, RFC 3198, November 2001
[2] TEQUILA IST project, http://www.ist-tequila.org
[3] CADENUS Project, “QoS Control in SLA Networks”, IST-1999-11017, March 2001
[4] R. Yavatkar, D. Pendarakis, R. Guerin, “A Framework for Policy Based Admission Control”, RFC 2753, January 2000.
[5] J.-P. Briot, Y. Demazeau, “Principles and architecture of MAS”, French book, Hermès, Paris, November 2001

[6] N.R. Jennings, M. Wooldridge, "Software Agents", IEEE Review, January 1996
[7] http://www.objectspace.com/Voyager
[8] http://www.trl.ibm.co.jp/aglets
[9] R. Ghanea-Hercock, "Agents: the future of intelligent communication", Intelligent Agents for Telecommunication Environments, HPS 2002
[10] RNRT ARCADE Project, URL: http//www.telecom.gouv.fr/rnrt/
[11] F. Krief, Z. Jzad, "An Intelligent User Interface for the Internet New Generation", ICTSM10, USA, October 2002
[12] T.M.T. Nguyen, G. Pujolle, N. Boukhatem N, "COPS usage for SLS negotiation", draft-nguyen-rap-cops- sls-00.txt, Work in progress, 2001

Service Provisioning and Management in Telecom Environments Using Active Network Technology

Fábio Luciano Verdi and Edmundo R. M. Madeira

Institute of Computing, University of Campinas (UNICAMP), Campinas-SP, Brazil
Telephone: +55 (19)-3788-5862
Fax: +55 (19)-3788-5847
{verdi, edmundo}@ic.unicamp.br

Abstract. The advent of Telecom over the last years brought up many challenges to service providers. The difficulty to offer services and, at the same time, to perform their management requires an integrated solution for both, service provisioning and management. In this paper, we outline an infrastructure to perform such tasks. This infrastructure allows to create Virtual Active Networks (VANs) and install services in Telecom environments based on the Active Networking technology. Besides service provisioning, the infrastructure performs some management functions taking into account mobile services. The management system is based on a mobile agent platform and considers the management of VANs, where services can migrate from a VAN to another. Some aspects about the implementation of a prototype we developed to test our approach are presented. In this prototype we focus on three specific management areas: accounting, performance monitoring and configuration.

1 Introduction

Service management in Active Networks (ANs) is currently a potential topic of research. Mobile Agent Technology (MAT) is a general name for facilities that support the transmission of code, as well as data, over a computer network. In this context, the AN technology is particularly attractive to be used in Telecom environments, due to its facility to send data and code to specific locations in the network. In the AN technology, the packets, also called *active packets* or *capsules*, can carry programs to be executed on routers and possibly change their state.

With AN technology, the customization of services can be achieved by partitioning the network into *Virtual Active Networks* (VANs), which customers can lease from service providers. A VAN can be described as a graph of virtual active nodes connected by virtual links where active packets can travel within a VAN or among different VANs [2]. The VAN concept is a very useful solution to isolate groups of customers and offer specific services to each customer's domain.

A possible solution to support the interaction between providers and customers is to develop a framework, which allows a customer to install, configure and run active services on his own VAN. These services can be static, or they can be moved from node to node within a VAN or among VANs. Information about accounting, performance and configuration enables the manager (human) to know the behavior of the managed environment to detect problems and take actions on them. In this paper, we describe a framework for service provisioning and facilities for policy-based management of these services. The services offered by the infrastructure are VAN creation and service installation, both implemented using the Active Node Transfer System (ANTS) [7].

Many approaches taken today have focused on AN technology. In this context, an architecture for network management that offers active services is presented in [3]. A model to manage mobile services in VANs is described in [5]. An approach for service management in Telecom environments is presented in [2]. The Mobile Agent paradigm and its use in the context of telecommunications is described in [4]. Taking into account the features from these approaches, our framework has incorporated some important aspects for Telecom environments such as: policy-based management, VANs, mobile services and mobile agent-based management. In contrast to [1], in which services are based on mobile agents, in our model the services are moved from a VAN to another using the active networking infrastructure. We have tested our approach on the Virtual Private Active Network (VPAN) service which is fully explained in [6]. An example of using such a service is when a group of researchers located in different companies needs to work together in a project, sharing resources and/or sending and receiving applications to be tested in different domains (e.g. integrated development of software).

The paper is organized as follows. Next section outlines our infrastructure and its components, and it illustrates four possible management scenarios. Section 3 describes some aspects about implementation and Section 4 presents the conclusion.

2 The Framework for Service Installation and Management

The framework we developed includes components to support VAN creation, service installation and management in virtual active Telecom networks. In this section we present these components, the defined policies, the configuration actions and, additionally, we focus on four possible Telecom scenarios which our model is able to handle. The management system we describe in this section is to manage a single domain. In [6] we address the extensions needed to support multi-domain management.

2.1 Policies

We have defined two possible solutions that can be realized when a client requires a service. The first solution considers that whenever a client needs a service, a new copy of this service will be created and sent to the client. The second alternative considers that whenever a client requires for a service, possibly a copy of this service from a node will migrate to attend the customer requirement. Both solutions are unfeasible whether their consequences are considered. To install a new service copy for each customer will flood the network with services. To migrate a service from a node to another to attend customer requirements will result in many migrations. Thus, we have tried an intermediate solution defining some policies to minimize both, the number of service copies and the number of migrations.

We created six representative policies:

1. There are three kinds of service: internal services which can only migrate in the same VAN, external services which can migrate among VANs, and VPAN services which can migrate among domains that belong to the same VPAN. The latter is explained in [6];
2. A service S has a limited number of copies in the network (**L1**): S_1,S_2,...,S_n, where n$\leq$**L1**;
3. A service S has a maximum bound of times to migrate in a period of time (**M1**). Each copy of S, S_i, has to follow this threshold;
4. The SP (Service Provider) can install a new service copy requested by a user until a threshold of **L2** (L2$\leq$L1).
5. The Least Recently Used Service (LRUS) will be the candidate copy to migrate to the destination host when a customer requires a new service. This happens if the threshold L2 was already reached;
6. A service copy may be removed if it does not migrate a minimum bound of times in a period of time.

Next, we present the cases that can happen considering these defined policies.

Analysis of Cases: Policies 2, 3, 4 and 5 defined above were created to implement our intermediate solution. There are four possible cases to be considered when a client requests a service. Let $\mathbf{S}_i$ to be the ith copy of service S, **C(S)** to be the number of copies of S in the network and $\mathbf{M(S}_i\mathbf{)}$ the number of migrations performed by S_i:

1. **C(S) < L2**: in this case the number of copies of S is less than L2 and, therefore, the SP can install a new copy of S. This minimizes the number of migrations;
2. **C(S) $\geq$ L2 and C(S) $\leq$ L1 and there is the LRUS, $\mathbf{S}_{lrus}$, with $\mathbf{M(S}_{lrus}\mathbf{)}$ < M1**: in this case, the threshold L2 was reached but the maximum number of migrations of S_{lrus} not and, therefore, S_{lrus} will migrate to attend the customer requirement. This minimizes the number of copies;
3. **C(S) $\geq$ L2 and C(S) < L1 and the LRUS, $\mathbf{S}_{lrus}$, with $\mathbf{M(S}_{lrus}\mathbf{)}$=M1**: in this case, the maximum number of migrations of S_{lrus} was reached, so this service will not migrate during a period of time, but it is possible to create a new service copy since C(S) < L1. During the interval from L2 to L1 there is a balance between creating new copies and migrating services;

4. **C(S) = L1 and the LRUS, S$_{lrus}$, with M(S$_{lrus}$) = M1**: in this case, the maximum number of migrations of S$_{lrus}$ and the maximum number of copies of S were both reached. The service will not be installed.

2.2 Configuration Management

We defined three types of possible actions to apply: create a new service copy (1), move a service for load balancing (2) and delete a service from the environment (3). The first one is responsible for creating a new service copy when cases 1 and 3 (as explained in Section 2.1) are considered. The second allows the manager to move services from a node to another in order to balance the load. The last one is useful to remove services which were created in order to attend a period of demand and no longer are being used.

2.3 Components of the Infrastructure

The infrastructure is composed of a Management Remote Application (MRA), a Management Center (MC), a Service Provider (SP), a Global Naming Service (GNS) and the Distributed Management Agents (Local Manager - LM and Managed Active Element Agent - MAEA) as described below (Fig. 1).

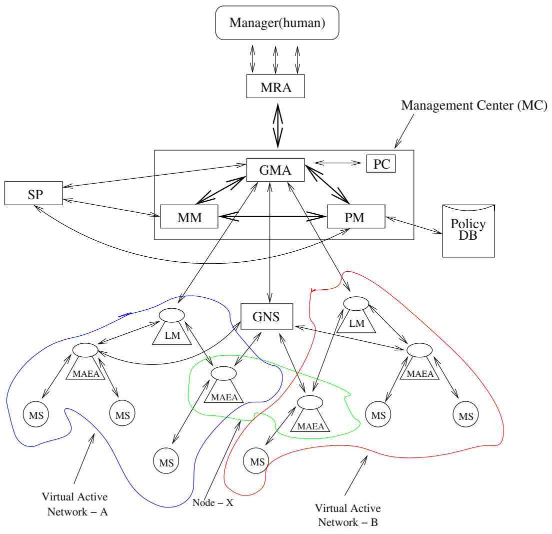


Fig. 1. Components of the Infrastructure and their relationship.

– **Management Remote Application - MRA**: This application presents to the manager (human) the interface with the methods related to the management questions (accounting, performance and configuration).

- **Service Provider - SP**: It is responsible for offering services to the customers. Clients send capsules to the SP to create a VAN and its services or to add a new service to their VANs.
- **Policy Manager - PM**: This component is responsible for controlling, insertion, updating and removal of a policy in the database.
- **Migration Manager - MM**: This component sends accounting management agents to the nodes and controls each service migration over the network applying the specific policies.
- **Global Manager Agent - GMA**: It is the centralized static manager of the model. There is only a GMA in a domain. This component is responsible for analyzing accounting, performance and configuration data on all VANs and their services.
- **Local Manager - LM**: There is an LM per VAN. It is responsible for the management of its VAN and collects data of each *Managed Active Element Agent* (MAEA, see below) located in the hosts of the VAN.
- **Managed Active Element Agent - MAEA**: It is responsible for managing one or more services in a host of a VAN, being the lowest level of the management. There is an MAEA per host per VAN. This agent is an interface between the ***Managed Service*** (MS) and the management system.
- **Global Naming Service - GNS**: This component receives the service location and the service identifier and creates a reference that indicates where the service is located.

Figure 1 shows two different VANs, A and B. We can see that node X belongs to both, but the model separates each one for service management.

2.4 Service Provisioning and Management Scenarios

The next four scenarios are associated with the customer's *Active Application* (AA), that is a customer's program to use the available environment. The first scenario describes in details the VAN creation and the mobile service installation. The second one presents a service installation following case 1 (see Section 2.1), i.e., without migration. The third scenario presents a mobile service migration between two VANs to satisfy a customer requirement. This scenario follows case 2 from Section 2.1 in which the threshold L2 was reached. The last scenario follows case 3 from Section 2.1 and it represents the situation in which the LRUS cannot migrate but a new service copy can be created. These four representative scenarios allow to demonstrate how our framework can be applied to offer VAN services and manage a typical Telecom environment.

1. **Creating a new VAN specifying what hosts and what services are required.**
   In this scenario the following sequence is needed, as shown in Fig. 2.
   Customer sends a capsule to the SP informing what hosts and what services each host will have (a); the SP sends the required services to the set of hosts (b); the SP notifies the MM and GMA about the new VAN (c); the

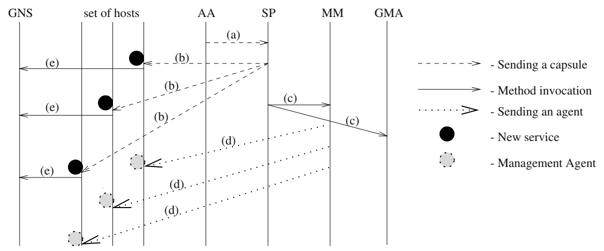


Fig. 2. Customer creating a new VAN.

MM sends accounting management agents to the hosts and creates the VAN manager (LM) (d); and finally services register themselves in the GNS (e). Next, we present the three possible scenarios to add a new service copy to the VAN. Figure 3 shows the first three steps (a-c) which are necessary to all of the three scenarios. These three steps are: customer sends a capsule to SP indicating what service is required and what destination host this service copy will be sent to (a); SP interacts with the MM in order to apply the policies on the required service (b); and MM gets the service policies from the PM to analyze them (c).

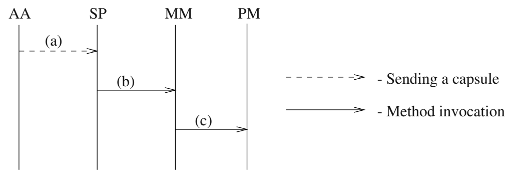


Fig. 3. Customer requiring a new service.

After performing these three first steps, one of the following three scenarios can be executed based on the results obtained by the SP.

2. **Adding a service to the VAN following case 1.**
   In this case, the threshold L2 has not been reached, so the SP installs a new service copy in the VAN to attend the client request. In addition to the first three steps from Fig. 3, the following steps are necessary (Fig. 4):
   SP sends a capsule to the destination host to create a new copy of the required service (d); SP notifies the GMA about the provisioning of a new service copy (e); the capsule arrives in the destination node, instantiates the new service copy, and registers it in the GNS (f); the service notifies its local MAEA that it has arrived (g); the capsule is forwarded to the new AA to notify that the required service has migrated. The AA can finally use the service (h).
3. **Adding a service to the VAN following case 2.**

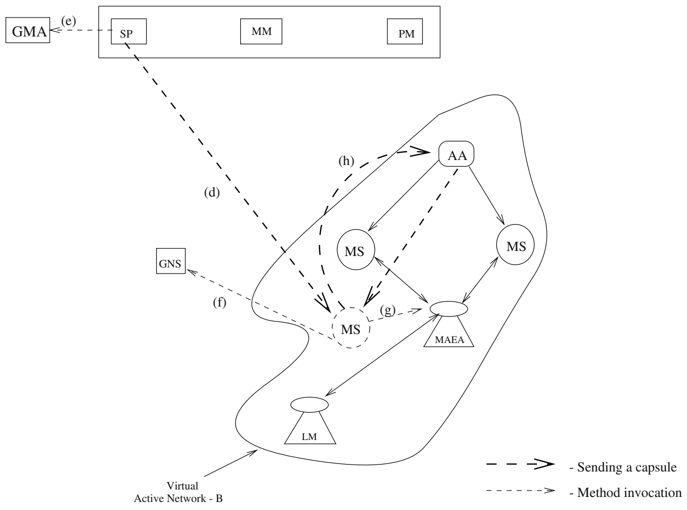


Fig. 4. SP installing a new service (case 1).

In this scenario it is considered that the limit L2 from policy 4 was achieved but the number of migrations not. If the required service is external, the LRUS can belong to another VAN and, then, the service must migrate between VANs. In addition to the three steps from Fig. 3, the following steps are necessary (Fig. 5):

SP gets the LRUS in the network. In this search, all the information about the LRUS is also obtained (d); SP interacts with the MM in order to verify whether the migration is possible or not (e); MM gets the service policies from the PM to analyze them (f); MM returns back the results to the SP (g); SP sends a capsule to AA of the VAN where the LRUS is located. This is necessary to notify the AA about the LRUS migration (h); SP notifies the GMA about the migration (i); the capsule is forwarded to the host where the service is located at this moment (j); the service unregisters itself from the GNS and notifies its MAEA that it is migrating (l); the source MAEA notifies the destination MAEA that a new service is arriving (m); the capsule removes the service from the local node and migrates it to the destination node (n).

To conclude the service installation, steps *f, g* and *h* from scenario 2 (Fig. 4) are executed.

4. **Adding a service to the VAN following case 3.**

   In this case, the LRUS cannot migrate during a period of time, but a new service copy can be created and sent to the client. In addition to the steps from Fig. 3, steps *d, e, f* and *g* from scenario 3 (Fig. 5) are performed to find the LRUS and apply the policies on it. After the SP realizes that the migration is not possible but a new service copy can be created, steps *d, e,*

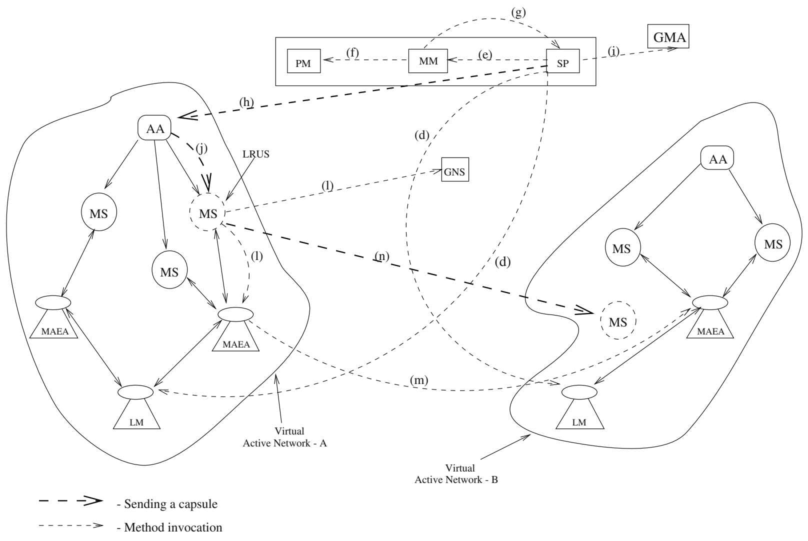


Fig. 5. Migrating the LRUS to attend the customer requirement (case 2).

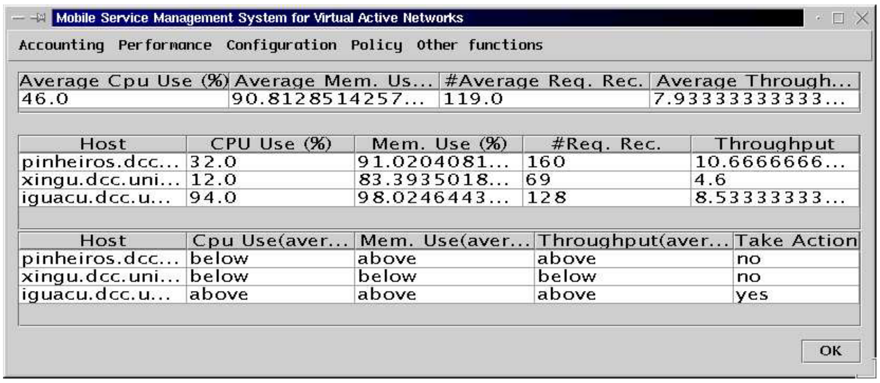
f, g and *h* from scenario 2 (Fig. 4) will be performed to conclude the service installation.

In scenario 3, each LM is responsible for finding the LRUS in its VAN and returns back the result to the SP. If the required service is internal (see policy 1), the same steps are performed, but in step *d*, SP obtains the LRUS only for the local VAN. For all other steps, the destination VAN is the local one.

3 Some Issues about the Implementation

To create the environment with VANs and services migrating among them, we used the ANTS toolkit. We have mainly used the following classes from ANTS: **Application**: in order to allow the customer to install and use services in the active nodes; **Capsule**: to send and receive code to and from the nodes; and **Node**: this class represents the execution environment of a VAN in a host. The active environment and the management system have been developed using Java 1.2. The services are implemented as Java objects. The mobile agent platform is the Grasshopper 2.1 [8], and the active nodes of each VAN are running on Sun Workstations executing SunOS 5.5. A service has a single name and is registered in the GNS. The way the service name is created is explained in [5].

The prototype allows to get management information in different levels: per service, per host, per VAN and per domain. Figure 6 shows a general view of the environment. It is possible to know, among other information, the throughput per host and the use of cpu and memory in each host.


Mobile Service Management System for Virtual Active Networks
Accounting Performance Configuration Policy Other functions
OK

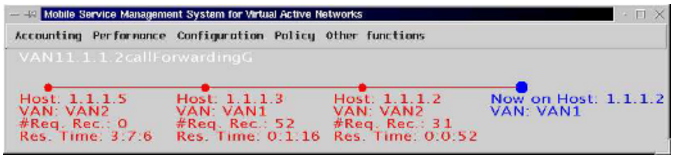
Average Cpu Use (%)	Average Mem. Us...	#Average Req. Rec.	Average Through...
46.0	90.8128514257...	119.0	7.93333333333...

Host	CPU Use (%)	Mem. Use (%)	#Req. Rec.	Throughput
pinheiros.dcc...	32.0	91.0204081...	160	10.6666666...
xingu.dcc.uni...	12.0	83.3935018...	69	4.6
iguacu.dcc.u...	94.0	98.0246443...	128	8.53333333...

Host	Cpu Use(aver...	Mem. Use(aver...	Throughput(aver...	Take Action
pinheiros.dcc...	below	above	above	no
xingu.dcc.uni...	below	below	below	no
iguacu.dcc.u...	above	above	above	yes

Fig. 6. Management Remote Application in a general view.

It is also possible to keep track of service migrations. Figure 7 depicts a service migration and some accounting information in each host. In this management operation, the service is named *VAN11.1.1.2CallForwardingG*. We can see that the service has migrated between VAN1 and VAN2 in order to attend the customer roaming. Initially, it is in the host 1.1.1.5 belonging to the VAN2. After some migrations, it is in the host 1.1.1.2 belonging to the VAN1. Also, below each host, we can see some information like the number of received requests and residence time.


Mobile Service Management System for Virtual Active Networks
Accounting Performance Configuration Policy Other functions

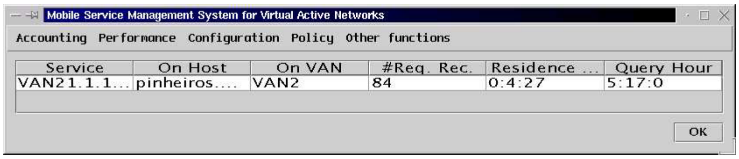
```
VAN11.1.1.2callForwardingG

Host: 1.1.1.5        Host: 1.1.1.3         Host: 1.1.1.2         Now on Host: 1.1.1.2
VAN: VAN2            VAN: VAN1             VAN: VAN2             VAN: VAN1
#Req. Rec.: 0        #Req. Rec.: 52        #Req. Rec.: 31
Res. Time: 3:7:6     Res. Time: 0:1:16     Res. Time: 0:0:52
```

Fig. 7. Service migration.

Figure 8 shows the most requested service in VAN2. We can see, among other information, the service name, the host where the service is located and the number of requests received by this service.

4 Conclusion

In this paper we presented a framework for service provisioning, such as VAN creation and service installation in Telecom environments based on VANs. The framework also performs the policy-based management of these services and their migration. The environment has a Service Provider, and a Policy Manager


Mobile Service Management System for Virtual Active Networks
Accounting Performance Configuration Policy Other functions
OK

Service	On Host	On VAN	#Req. Rec.	Residence ...	Query Hour
VAN21.1.1...	pinheiros....	VAN2	84	0:4:27	5:17:0

Fig. 8. The VAN2 most requested service.

responsible for controlling and processing policies. The proposed management model is based on a mobile agent platform and considers three management functional areas: accounting, performance and configuration. This model is flexible in the sense of management questions can be answered from different levels: per service, per host, per VAN, and per domain.

We create some scenarios to simulate a Telecom environment: exploiting the VAN creation, installing mobile services, and managing a service migration. The policies we defined gave us a good comprehension on how our intermediate solution is useful to minimize the number of service copies and the number of migrations. The implementation has showed that the proposed framework can handle Telecom environments with mobile services and VANs successfully, opening up a potential further study.

Acknowledgements

The authors would like to thank CNPq and CAPES for their support.

References

1. M. Breugst and T. Magedanz. Mobile Agents - Enabling Technology for Active Intelligent Network Implementation. *IEEE Network,* pp. 53-60, May/June 1998.
2. M. Brunner, B. Plattner, and R. Stadler. Service Creation and Management in Active Telecom Networks. *Communications of the ACM,* pp. 55-61, March 2001.
3. I. W. Marshall, H. Gharib, J. Hardwicke, and C. Roadknight. A Novel Architecture for Active Service Management. *Seventh IFIP/IEEE International Symposium on Integrated Network Management,* Seattle, Washington, USA, May 2001.
4. V. A. Pham and A. Karmouch. Mobile Software Agents: An Overview. *IEEE Communications Magazine,* pp. 26-37, July 1998.
5. F. L. Verdi and E. R. M. Madeira. A Mobile Agent-based Model for Service Management in Virtual Active Networks. *IFIP/IEEE 12th International Workshop on Distributed Systems: Operations & Management - DSOM'01,* Nancy, France, pp. 101-112, October 2001.
6. F. L. Verdi and E. R. M. Madeira. Service Provisioning and Management in Virtual Private Active Networks. *IEEE 9th International Workshop on Future Trends of Distributed Computing Systems - FTDCS'03,* San Juan, Puerto Rico, pp. 205-211, May 2003.
7. D. J. Wetherall, J. V. Guttag, and D. L. Tennenhouse. ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols. In *IEEE OPENARCH'98*, San Francisco, CA, April 1998.
8. IKV++ GmbH - Grasshopper: http://www.grasshopper.de.

ASPOSE: Agent-Based Service Provisioning in Open Services Environment

Xin Li, Zhengkun Mi, and Xudong Meng

Department of Communication Engineering
Nanjing University of Posts and Telecommunications, 210003, P.R. China
{d0105, mizk, mengxd}@njupt.edu.cn

Abstract. Intense competition in telecommunication industry together with ever-increasing user requirements is changing service provisioning pattern in recent years. It is conceivable that future telecommunication services will be offered by service providers over network infrastructure owned by independent network providers. Consequently, service providers and network providers must cooperate to satisfy the user requirements, such as provision of personalized services across network and terminal boundaries with the same look and feel. To satisfy those requirements, an open service provisioning framework that facilitates seamless service provisioning across multi-domain networks is needed. In this paper, we propose such a service provisioning framework which is called ASPOSE (Agent-based Service Provisioning in Open Services Environment). We describe the framework in detail ranging from user terminal registration, service deployment to service execution. In the system implementation aspect, we apply an iterative design process in which TINA (Telecommunication Information Networking Architecture) is adopted as the system modeling template and UML (Unified Modeling Language) as the description language.

1 Introduction

In recent years, due to ever-increasing competition in telecommunication industry, increased effort has been made to open up telecommunication networks to facilitate flexible service provisioning. It is expected that future telecommunication networks will consist of complex relationships among various stakeholders such as network providers, service providers and end users. With the open-up of telecommunication network, the intense competition for customers is driving service providers to offer advanced services at accelerating rate. For example, The VHE (Virtual Home Environment) concept implies the user to be consistently presented with the same personalized features, user interface customization and services in whatever network, with whatever terminal (within the capabilities of the terminal), and wherever the user may be located [1][2]. At the same time, context-aware service requires network to provide user with personalized service related to user's current situation such as location, time and activity [3].

To manage such a complex service provisioning environment, an open service provisioning framework that supports flexible user and service management is imperative. This paper proposes such a framework which is called ASPOSE (Agent-based Service Provisioning in Open Services Environment) through utilization of mobile agent technology.

Mobile agent represents state-of-the-art distributed programming paradigm, and many researches have worked on the application of mobile agent in various fields such as information searching and network management. Mobile agents are dynamic, asynchronous, and autonomous, making the programming paradigm suitable for developing distributed system for mobility-enabled services. By using mobile agent technology, we expect that the framework can be designed in a succinct and flexible manner.

This paper is organized as follows. In section 2, we review some related projects and our previous work with the contribution of our work mentioned. In Section 3, we present the ASPOSE architecture with a detailed discussion of its components. In section 4, the service provisioning mechanisms in ASPOSE framework are illustrated, including user registration, service deployment and service execution. In section 5, we explain the design methodology adopted in the implementation of ASPOSE system. Section 6 concludes this paper with a mention of future work to be done.

2 Related Work

Software agent has been identified as a promising approach to simplifying the design and implementation of complex telecommunication networks. Most of the research efforts, however, are concentrated to network management and information retrieving. Limited work is concerned with the application of agent to service provisioning, among which MARINE project [4] is the pioneer exploring mobile agent based Intelligent Network (IN) enhancement. In line with it, our previous work proposed phased evolution approach of current IN towards mobile agent-based distributed service provisioning framework [5]. Unfortunately through years of study, it is realized that industry has little interest in introducing new technology in the matured IN and regards IN as a service provisioning solution dedicated to circuit switched PSTN/ISDN. People turn their enthusiasm into the innovation of value added service provisioning in Next Generation Network (NGN). Therefore we consider that it is more practical to investigate the application of mobile agent technology to the future NGN service provisioning, with focuses on IP and mobile environment. Inspired by this thought, we initiated the research on ASPOSE. Its key objectives are as follows:

- To study the application of mobile agent technology in service provisioning field.
- To exploit synergy of mobile agent and other distributed computing technology such as EJB (Enterprise Java Bean) in developing distributed system.
- To study development methodology of agent-based distributed service provisioning system.
- To demonstrate the execution of VHE service in the framework.
- To demonstrate the execution of context -aware service in the framework.

It was noticed that some EU projects also laid their attention in this regard, among which there are VESPER [6], CAMELEON [7] and Project P810 [8]. The former aimed at an enhanced layered structure to better support VHE. Unfortunately it failed to reach a final result. The latter two were related to mobile agent application to mobile network service provisioning but with little concern on context-aware requirements and multi-terminal usage situation in the context of multi-domain environment. Taking these observations into account, we proposed an agent-based distributed service provisioning framework [9] with reference to TINA (Telecommunication Information Networking Architecture) model. This paper results from the further work based on [9], presenting the detailed ASPOSE architecture. In addition, the design of ASPOSE system is also briefly discussed, incorporating useful principles and experiences for the distributed system design revealed in Prospect Project [10].

The contributions of our work include:

- To present a practical service provisioning environment in which multiple domains, i.e., network provider domain, service provider domain and end user domain, are differentiated according to their respective roles.
- To describe how these domains are coordinated to provide user with advanced services, such as VHE services and context-aware services, by using agent technology.
- To propose the design methodology in implementing such an agent-based system by use of TINA model combined with UML (Unified Modeling Language) as the description language.

3 Architecture Design

3.1 ASPOSE Architecture

Our project is related to the design and implementation of an integrated open framework for the support of flexible service provisioning, charging, and location information management by using mobile agent technology. The ASPOSE framework intends to support functions capable of transparently and dynamically adapting services to client location, user preferences, and device characteristics. Figure 1 illustrates the ASPOSE architecture proposed in our project.

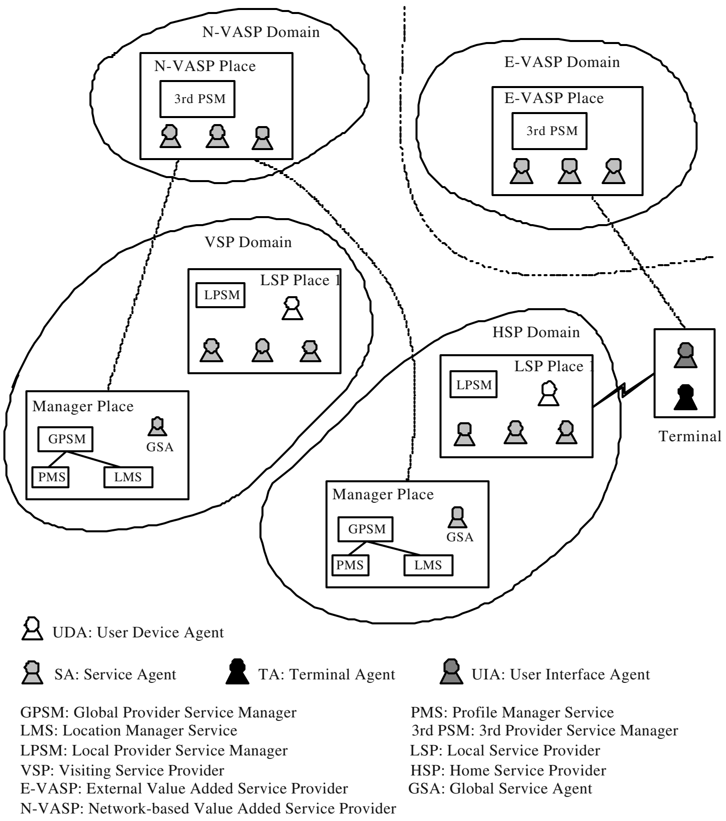

N-VASP Domain
N-VASP Place
3rd PSM
E-VASP Domain
E-VASP Place
3rd PSM
VSP Domain
LSP Place 1
LPSM
HSP Domain
LSP Place 1
LPSM
Terminal
Manager Place
GPSM
GSA
PMS
LMS
Manager Place
GPSM
GSA
PMS
LMS
UDA: User Device Agent
SA: Service Agent
TA: Terminal Agent
UIA: User Interface Agent
GPSM: Global Provider Service Manager
PMS: Profile Manager Service
LMS: Location Manager Service
3rd PSM: 3rd Provider Service Manager
LPSM: Local Provider Service Manager
LSP: Local Service Provider
VSP: Visiting Service Provider
HSP: Home Service Provider
E-VASP: External Value Added Service Provider
GSA: Global Service Agent
N-VASP: Network-based Value Added Service Provider

Fig. 1. Architecture of ASPOSE

To a specific user, the architecture presents four domains, i.e., home service provider (HSP) domain, visiting service provider (VSP) domain, network-based value added service provider (N-VASP) domain, and external value added service provider (E-VASP) domain.

HSP domain is the place where user subscribes to services and stores his information, such as user profiles, user terminal profile and user location information. VSP domain is the place where the visiting user can access service on condition that the VSP domain has federation agreement with the user's HSP domain. N-VASP is a 3rd service provider that registers its services to a HSP to make its services visible to the users in the HSP domain. E-VASP is only related to users that subscribe service from it. From the viewpoint of service provisioning, E-VASP is invisible to HSP domain. E-VASP uses connection function provided by network provider.

3.2 Roles of Agent and Domain Component

In ASPOSE, four kinds of agents are defined. Among those, user device agent (UDA) represents user terminal at network side; service agent (SA) encapsulates service logic provided by network providers or service providers; user interface agent (UIA) is an interface agent for a specific terminal related to a certain SA; and terminal agent (TA) represents terminal capability in the user terminal. In the following, we will give further descriptions of these agents in the context of explanation of domain components within the architecture.

HSP domain belongs to a specific network provider, while a network provider can have more than one HSP domain. An HSP domain forms an autonomous region that provides a centralized management of its users and users' terminals as well as services provided in this domain. In addition, through cooperation with other VSP domains, it can provide advanced services such as VHE service to users.

HSP domain is composed of one manager place and several local service provider (LSP) places.

3.3 Manager Place

The manager place is in charge of service management, including service registration, discovering and SA control. The manager place is also responsible for user information management, such as user location management, user profile management and user terminal profile management. Furthermore, manager place accommodates global service agent (GSA), which implements domain-wide service. The core component in manager place is global provider service manager (GPSM), whose functions can be summarized as follows:

- Together with PMS (Profile Manager Service) and LMS (Location Manager Service), it manages user information.
- Handling service registrations from N-VASP.
- Discovering services for end user according to user and terminal profile.
- In charge of locating UDA in response to request from HSP domain's LSP place or VSP domain's manager place.

Other components in the manager place are:

- PMS: A registrar of domain's user profiles and user terminal profiles.
- LMS: A registrar of user location information. It may translate terminal physical address to service-related location so as to provide users with context-aware services.

3.4 LSP Place

In each HSP domain, LSP Place corresponds to a specific operator network, such as mobile network or fixed PSTN. LSP Place provides execution environment for SA.

In ASPOSE, from network access prospective, two kinds of SA are distinguished: NCSA (Network Capability Service Agent) and service-specific SA. NCSA is a common SA that negotiates network capabilities and resources on behalf of UDA and service-specific SA. For example, prior to the execution of a service-specific SA, the SA may need to invoke NCSA to decide whether the network can provide necessary resources.

In an LSP Place, the core component is LPSM (Local Provider Service Manager), which deals with the following functions:

– Responsible for the management of local service-specific SAs.
– Instantiation of service-specific SA upon receiving request from UDA.
– Responsible for user terminal registration when a user accesses to the network, i.e., generates a UDA for the terminal if the UDA does not exist in the network, or requests the existing UDA to migrate to current place if the terminal's UDA exists in another place.

3.5 Relationship between User, Terminal, and Service

In ASPOSE, a user belongs to a specific HSP domain, and all of the information concerning this user is managed by the domain's manager place. A user can have several terminals with different characteristics. ASPOSE allows multiple terminals registered in the network, and those terminals can span in different network providers.

The user profile is managed by the user's home GPSM. A user can have several user profiles contained in the PMS. But at one time, only one user profile is active. The active user profile may be determined by GPSM or manually selected by the user.

For each terminal there exists a UDA, which is under control of the terminal's home GPSM. UDA always follows its terminal and migrates to the network the terminal currently resides in.

In the terminal, there exists a TA for each terminal. TA is stationary agent and it will remain in the terminal forever. The functions of TA are:

– Interacts with end user, e.g., display a GUI to its user.
– Interacts with network-side UDA corresponding to this terminal.
– Accommodates UIA to provide service-specific interface to the user.
– Provides personal assistant functions, such as scheduling a meeting for its user according to information it gathered.

UIA is an interface agent that can migrate to the user's terminal to present service-specific interface to the user. Each SA has several UIAs to cater for different user terminal capabilities. SA decides on which UIA should be migrated to terminal by using information got from terminal's UDA. Detailed discussion is covered in the following sections.

In sum, in ASPOSE each user has several user profiles with one active each time instant; each user has several terminals, and each terminal has a UDA in the network when the terminal has registered to the network. The coordination of those UDAs is handled by the user's home GPSM according to user profile. To each executing SA, a specific UIA belonging to this SA resides in a user terminal.

4 Service Provisioning Mechanisms

4.1 Terminal Registration

Before requesting service execution, a user (his terminal) must register to a serving network. As illustrated in Figure 2, the terminal belonging to HSP Domain roams from its home Domain to LSP Place 1 belonging to VSP Domain. Supposing that the terminal's UDA already exists in the network, the registration process can be described as follows:

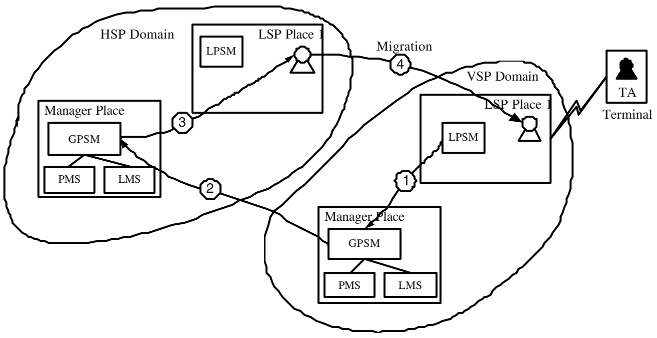


Fig. 2. Terminal inter-domain registration

Once LPSM in LSP Place 1 belonging to VSP Domain detects access of a terminal, it requests UDA information concerning the terminal from VSP Domain's GPSM (1). According to GUID (Global Unique Identity) contained in the registration information, the request in step (1) is forwarded to HSP Domain's GPSM (2): GUID contains identity of the terminal's home domain. Subsequently, GPSM in HSP Domain inquires terminal information from its LMS and knows that the UDA corresponding to the terminal is in LSP Place 1 belonging to HSP Domain. Thereafter, a request is issued to the UDA and informs it to migrate to LSP Place 1 of VSP Domain (3). According to the request, the terminal's UDA migrates to LSP Place 1 of VSP Domain (4). Hereafter, the UDA gets terminal profile and interacts with its home domain's GPSM to get service list. Moreover, UDA also gets service list from visiting place's LPSM. For the sake of simplicity, these interactions are not shown in figure 2.

4.2 Service Deployment

As discussed in the former section, two kinds of service provider are distinguished in ASPOSE: network service provider that provides network access functions as well as some basic services, 3rd service provider which has no access network infrastructure but only provides value added services either directly or indirectly to users.

In ASPOSE, all services are modeled as service agents, which encapsulate specific service logics. The network providers' services are deployed in manager place and

LSP place, and among these SAs provided by network service providers, GSA and LSA (Local Service Agent) are distinguished. GSA is deployed in the manager place that is domain-wide available, whereas LSA is deployed in LSP place that is only available in the LSP place. On the other hand, 3rd services are deployed in N-VASP or E-VASP. It is worth noting that herein we only discuss service layer, and as to how SAs are mapped to underlying network functions is tackled by network APIs (Application Programming Interfaces) such as OSA (Open Service Access) framework.

LPSM and GPSM provide web-like open API to facilitate convenient and automated service deployment. Through the open API 3rd service providers and network service providers can register their service agent information to the platform, as well as to delete and update existing SA descriptions. During service deployment process, SA provides LPSM/GPSM with an XML document including all the necessary information for supporting provisioning of the service to end users. The XML document includes the service agent identifier, as well as data needed for searching, accessing and managing the service. Figure 3 illustrates the N-VASP service registration process. Note that the 3rd PSM acts on behalf all the SAs accommodated by the N-VASP place to register service description information to GPSM belonging to an HSP Domain.

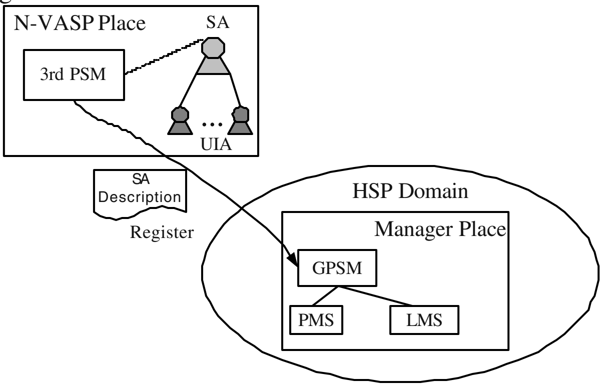


Fig. 3. Service deployment in N-VASP

4.3 Service Execution

Once a terminal has registered to the network, the user can select a service to execute from the service list presented in his terminal. Since the terminal registration process has been discussed in previous paragraph, the following discussion supposes that the terminal's UDA exists in the network where the terminal is in.

In the following, we will discuss a service execution scenario where a user requests a service form visiting domains, while the SA encapsulating the request service logic is located in the user's home domain.

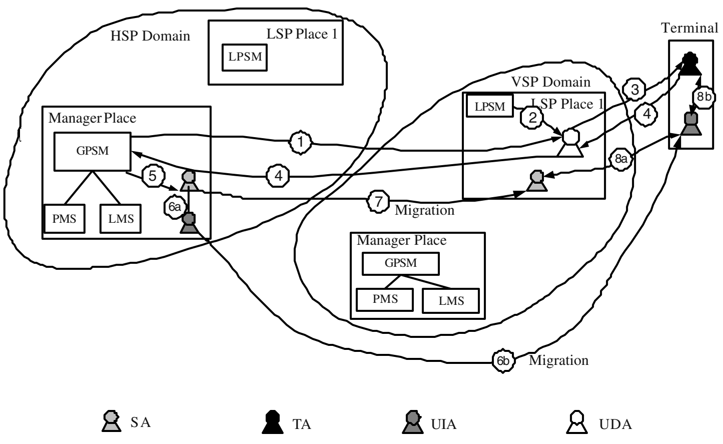


Fig. 4. Network provider service execution in visiting domain

As depicted in Figure 4, the terminal belonging to HSP Domain roams from its home domain to LSP Place 1 of VSP Domain and requests service execution. In this context, the terminal can execute visiting domain's services as well as home domain's services. Supposing that the requested service's SA resides in the terminal's home manager place as GSA, and the GSA decides to migrate to the terminal's current network location before execution. The service execution process can be described as follows:

1. The terminal's UDA gets service list from its home GPSM.
2. UDA additionally gets service list from visiting place's LPSM.
3. UDA transfers service list to the terminal to display to the user.
4. User chooses a service from the service list, and the selection is notified to UDA. Since UDA contains the selected service's provider location information, this request is forwarded to HSP Domain's GPSM.
5. HSP Domain's GPSM knows that the requested service's SA resides in the manager place. It then instantiates a GSA and transfers the user's service-related information to the GSA.
6. According to user profile and terminal profile, the GSA selects and instantiates a UIA for the terminal and informs it to migrate to the terminal.
7. GSA migrates to LSP Place 1 of VSP Domain according to its reasoning result.
8. UIA interacts with user and the GSA to aid the execution of the service.

5 Design Methodology

In order to assist the design and implementation of ASPOSE system, we have developed a design process in which standard modeling and description techniques are applied. Since mobile agent technology can be regarded as extension of object-oriented technology, object-oriented design techniques can be utilized in the design of agent-based system. UML is the de-factor modeling language for object-oriented system description, which is independent of any software development process. UML extensions for agent modeling are also proposed, which will further facilitate the design of agent-based system by using existing techniques.

By its nature, ASPOSE system requires a solution for system description to tackle the inherent complexity in such a multi-domain environment. Currently several standards address the issue of specifying open system. TINA provides a suitable model that fits to our needs, which builds on ODP (Open Distributed Processing) standard. By suggesting five different viewpoints (Enterprise, Information, Computation, Engineering and Technology), ODP provides a basis for the separation description and discussion of different aspects of the design and implementation of the system.

For the discussion given above, we apply an iterative design process in which TINA is adopted as the system modeling template and UML as the description language. The ASPOSE design process is concerned with the analysis, development, implementation and trial of ASPOSE system in an incremental way. The process is iterative and system functions need not to be addressed in a single iteration cycle. Figure 5 depicts the design cycle of the design process adopted by ASPOSE. The cycle shows the artifacts generated in each design stage in accordance with UML specification.

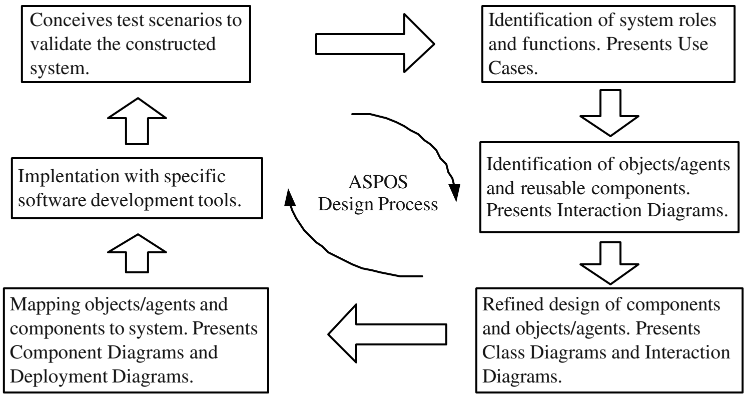

Conceives test scenarios to validate the constructed system.
Identification of system roles and functions. Presents Use Cases.
Implentation with specific software development tools.
ASPOS Design Process
Identification of objects/agents and reusable components. Presents Interaction Diagrams.
Mapping objects/agents and components to system. Presents Component Diagrams and Deployment Diagrams.
Refined design of components and objects/agents. Presents Class Diagrams and Interaction Diagrams.

Fig. 5. ASPOSE design process

Using reusable software components to construct distributed systems is a key design strategy in TINA, which resulted in a flexible and consistent system. In the context of ASPOSE system design, a number of reusable software components are designed, i.e., ISMC (Integrated Session Management Component), SMC (Subscription Management Component), and AMC (Accounting Management Component). ISMC is responsible for session management among user, terminal, service provider, and service provider manager; SMC acts on behalf of service provider in dealing with user service subscription, service registration between service providers and service management functions used by end users and service provider managers; AMC provides service providers with billing as well as system operation logging facilities. Detailed discussion of ASPOSE system design will be given in a later publication.

6 Conclusion and Future Work

Realized that software agent is a promising design paradigm in the development of future telecommunication networks, this paper proposed the ASPOSE framework. The main objective of this framework is to design and implement a demonstration service provisioning system aiming to study the application of software agent technology in telecommunication field.

ASPOSE framework is currently under construction stage. In the implementation of ASPOSE system, we choose to integrate mobile agent platform with J2EE (Java 2 Enterprise Edition) platform to exploit synergy of mobile agent and state-of-the-art component technology in designing such a distributed system. In the process of system design, we have found that software agent abstraction simplifies system management as well as service control. However, how to encapsulate service logic in SA with flexibility yet easy to manage is a challenging issue. To tackle this problem, a design method that guides the development of service agents is under investigation. As some concrete services are imp lemented and tested in this framework, we will further evaluate the performance and effectiveness of the proposed framework. And in this way, we can know what implications such a novel service provisioning framework will impose upon underlying networks fro m an engineering point of view.

References

1. 3GPP Third Generation Partnership Project. http://www.3gpp.org/
2. Paolo Bellavista, Antonio Corradi, and Cesare Stefanelli. The Ubiquitous Provisioning of Internet Services to Portable Devices. PERVASIVE computing, pages 81-87. 2002.
3. Spyridon Panagiotakis and Athanassia Alonistioti. Intelligent Service Mediation for Supporting Advanced Location and Mobility-aware Service Provisioning in Reconfigurable Mobile Networks. IEEE Wireless Communications, pages 28-38. October 2002.
4. ACTS MARINE Project. http://www.italtel.it/drsc/marine/marine.htm
5. Fang Fang, Mi Zhengkun: "Strategy of Evolution towards Mobile Agent-based Distributed Intelligent Network", In International Conferences on Info-tech and Info-net Proceedings (ICII2001), pages 747-752, Bejing, China, Oct 2001.
6. IST VESPER Project: "VHE Requirements", Public Deliverable D21, Nov.2000.
7. CAMELEON Project. http://www.comnets.rwth-aachen.de/~cameleon/
8. EURESCOM Project P810. http://www.eurescom.de/public/projectresults/P800-series/800s.asp
9. Chai Yawei and Mi Zhengkun, "An agent-based distributed service architecture for next generation network", ICT'02 Proceedings, pp.356-359, June 2002.
10. Prospect Project. http://www.fokus.fraunhofer.de/research/cc/platin/coop/prospect/

Policy-Based Service Provisioning and Users Management Using Mobile Agents

Mohamed Ganna, Eric Horlait

Université Pierre et Marie Curie
Laboratoire LIP6-CNRS
8, rue du capitaine Scott 75015 Paris, France
{mohamed.ganna, eric.horlait}@lip6.fr

Abstract. Many papers have addressed policy-based management of networks focusing on security and QoS. However, users' mobility remains a difficult aspect of this approach. There is considerable difficulty in supporting the provision of QoS and security services in a common architecture to satisfy the many requirements that a user can ask for. The need for automated management shows the necessity to use languages to define policies. These languages are still not sufficient since they don't provide tools to map high-level policies into device-dependant ones or they offer only the possibility to cover either management policies or security ones. We present in this paper a policy-based management architecture for the provisioning of services to users using mobile agents. We based this architecture on the notion of a domain. Each domain represents an administrative authority with its own behavior, polices, and users. The agents are under the control of a set of policies that define what to do when a mobile foreign user connects into the domain. When a mobile user is visiting a domain and asks for a service, this domain supplies him with a negotiated QoS and security. The negotiation includes the mobile user and the foreign domain, or the home, the foreign and eventually neighboring ones. The user's profile is a starting point of the negotiation. This agent-based architecture supports mapping polices from high level to network ones. To do so, different agent collects information concerning the devices' capabilities and the technologies supported.

1 Introduction

Secure data transmission, resource allocation, and user's mobility need an effective management to meet their entire requirements. However, there are only dedicated partial solutions for these general problems. Policy-based management offers a good means for an automated and centralized control of administrative domains. Each domain is under the authority of a unique entity called PDP (Policy Decision Point) that controls and manages different PEPs (Policy Enforcement Point), representing the architecture proposed by the IETF (Internet Engineering Task Force). The PDP is responsible for the processing and management of all events in the domain related to resource allocation. The basic key action is initiated by either client queries or resource management constraints. This solution, even if it seems realistic, is still incomplete because the management policies are different from one domain to another and becomes ineffective with mobile users visiting other domains. Communication involves the use of resources from multiple domains, the management of mobile users and mobile environment. Thus, the big issue is the QoS, security and mobility management for the intra and inter domain service provisioning. Published works present partial approaches dealing only with QoS, security or mobility, or two of the three features but none focuses on integrated approach of all of them. Our claim is that a policy-based model is better for managing the relationship between all concurrent constraints. Policies are the rules that dictate the behavior of the system. Mobile agents offer also a good deal for real time service establishment and for wireless environment that presents many bandwidth and connectivity constraints. Mobile agents have the advantage to avoid multiple moves in the negotiation phase as they convey all the necessary information. Their use requires a security consideration and needs a choice of a mobile agent platform that offers security features for mobile agents and the execution environment.

This paper is presented as follows. Section 2 presents a state of the art of policy description languages. Section 3 gives an overview of some interesting architectural models dealing with QoS, security or mobility management. Section 4 gives a definition of mobile agents, and introduces some mobile agents' approaches. Then architecture based on mobile agents and policies managing the users, their mobility and providing them with QoS and security will be presented. In section 6 a negotiation scenario is given to show the working of the system. Finally, a last section describes our future research directions and presents some concluding remarks.

2 Policy Specification Languages

A policy is a rule that defines the behavior of a system [1] beyond strict technical aspects. These rules are defined by the domain administrator. In this respect, among other methods, a specification language can be used. Policies are expressed as rules applied to objects through conditions, constraints and events. Policy languages must support possible conflict and inconsistencies analysis in the specification. Extensibility is also needed to cater for new types of policies. The language must also be easy to use and comprehensible by users and provide mapping mechanisms so that policies can be understandable by network devices.

Few works on policy specification relate to QoS. They are based on scripting or interpreted language approaches (TCL or Java). Security is the main concern of most researches. Approaches based on formal methods are not intuitive and the mapping onto implementation mechanisms is not very easy. Their understanding is quite difficult because they assume a strong mathematical background. ASL (Authorization Specification Language) [2] is a formal logic language for specifying access control policies including a Meta-policy called integrity rules. This latter limits the range of acceptable access control policies by specifying an application-dependant set of rules. It also provides support for role-based access control. But it doesn't offer method for grouping rules so it doesn't scale well for large systems. There is also no way for specifying authorization rules for groups of object. Ortalo [3] expresses security policies based on the logic of permissions and obligations. The user must define what is permitted and what he must do. Ortalo is not suitable for modeling authorization and obligation policies, as it doesn't distinguish the two concepts. Lasco [4] offers a graphical method to specify security constraints. But the scope of this approach is limited to security requirements. The Policy Description Language (PDL) of the Bell Labs is an event-based language [5]. Policies use rules in the form of event-condition-action to map a series of events into actions. However PDL doesn't support the composition of policy rules into roles.

Ponder [6] offers a common means of specifying management policies to determine what actions are carried out when specific events occur within the system or what resources to allocate under specific conditions. Security policies are specified using role-based access control. It is a declarative, object-oriented language, flexible and extensible to cover the requirements of distributed systems. It has four basic policy types: authorizations, obligations, refrains and delegations and three composite policy types: roles, relationships and management structures used to compose policy. Ponder defines domains for grouping managed objects, events for triggering obligation policies and constraints for controlling the enforcement of policies. Each obligation policy, defining what actions must be done, is related to authorization policies that determine the rights given to the object enforcing policies. Ponder offers a good tools and a syntactic form for specifying and managing policies.

The literature about policy specification languages is extremely rich. Other works on specifying languages for the management of QoS can be found in [7] [8]. Also a comparison of the different security and QoS languages is conducted in [9]. Table 1 gives a comparison of policy specification languages based on some criteria that a general policy specification language should have [10]. We compare the languages introduced above and the following:

– RDL: the Role Definition Language is a role specification language developed for secure internetworking services.
– SPL: the Security Policy Language is an event-driven policy that supports access control, history-based and delegation policies.
– HQML: Hierarchical QoS Markup Language is an XML based language. Its goal is to enhance the distributed multimedia applications.

Table 1. Comparison of policy specification languages

Languages	TCL/Java	Ortalo	RDL	SPL	LaSCO	PDL	Ponder	ASL	HQML
Authorization Policies	√	√	√	√	√	√	√	√	
Delegation Policies	√	√	√				√		
Constraints Supported					√	√	√		
Scalability	√		√	√		√	√		
Structuring			√	√			√		
Reuse		√	√	√		√	√		√
Conflict Analysis		√					√	√	√
Refinement	√					√			√
Management Policies	√		√			√	√		√

As we can conclude from this table, Ponder offers a good alternative for specifying QoS and security policies. It provides a well-defined representation and grouping of objects in the system. However, some features must be added for the mapping and feedback mechanisms. PDL is another option having refinement mechanisms and supporting security and management policy specification.

3 Management Architecture

Most research on Policy Based Network Management (PBNM) focus only on intra-domain QoS and security. There is no much work on a policy-based management of mobility. We can deduce that only partial approaches are available, addressing the inter-domain management and the provisioning of services. The service provisioning to mobile users is done with an a priori negotiation of the users' preferences represented by profiles stored in the users' home domain. We present architectures where each domain can represent its policy rules, allowing negotiation and enforcement in the inter-domain area.

3.1 QoS Architecture

The inter-domain QoS provision is concerned by performance guarantees and reservation capabilities for the end-to-end communication. When a user asks for some bandwidth reservation, technical as well as political aspects are to be taken into account. This becomes quite complex when more than one administrative domain is concerned. Many works have proposed solutions to this problem by providing centralized architectures with an entity responsible for establishing the end-to-end QoS provisioning.

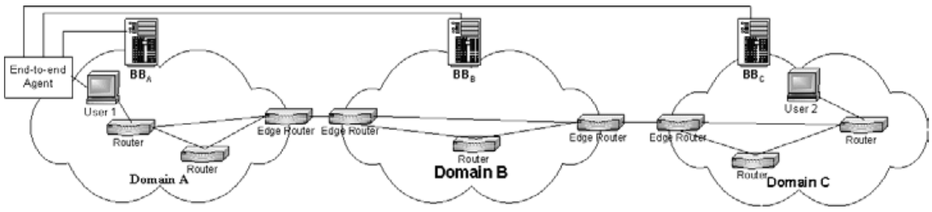


Fig. 1. Source-domain-based signaling is controlled by a source domain entity that contacts all related BBs directly

[11] presents architecture with a Bandwidth Broker (BB) in the administrative network domain. A BB provides admission control and configures the edge routers of a single domain. It is responsible for finding a path between the two communicating hosts and negotiates the QoS needed with the intermediate Bandwidth Broker. A Service Level Agreement (SLA) between peered domains regulates the acceptance and the constraints of a given traffic profile. A network reservation for traffic traversing multiple domains must obtain multiple network reservations. To accomplish this, two approaches are proposed. In the first, called "Source-Domain-Based Signaling", a user of the source domain or an agent working on behalf of it, contacts each BB individually (Figure 1). A positive response from every BB

indicates that the user has an end-to-end reservation. The problem is that each BB must know about the user in order to perform authorization.

In the second approach called “Hop-by-Hop-Based Signaling” all the reservation requests are propagated between BBs rather than all originating at the end-domain. User1 in domain A (Figure 2) contact BB_A, which propagates the reservation request to BB_B only if the reservation has been accepted by BB_A. Similarly, BB_B contacts BB_C. So each BB needs to know only about its neighboring BBs, and all BBs are contacted. The source-domain-based signaling is faster than the hop-by-hop one, because the reservations for each domain can be made in parallel.

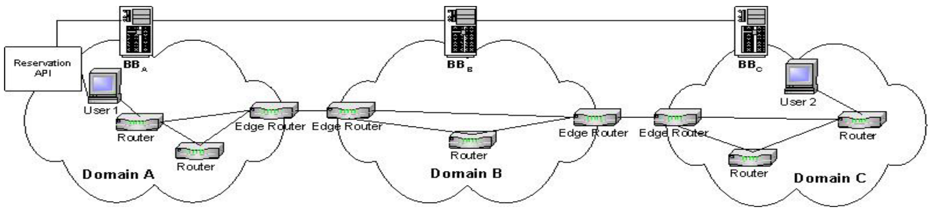


Fig. 2. Hop-by-hop signaling is done using an authenticated channel between peered BBs among the downstream path to the destination

[12] proposes, with its architecture of policy framework, two protocols: a Policy Advertising Protocol (PAP) that distribute local policies among domains, and a Policy Negotiation and Notification Protocol (PNP) that sets up a QoS-guaranteed communication path among domains. PAP is an enhanced BGP4 (Border Gateway Protocol-4) protocol and PNP a COPS (Common Open Policy Service) enhanced one. Policy exchange between domains provides information about which services are available in another domain, which customers exist in another domain, how much resources are available in another domain, and additional information such as charging tables for services and resources in another domain. Such information helps to find the communication context in the communication path.

3.2 Security Architecture

Security problem is also a subject of research and many projects are developing management security architecture. But this has created heterogeneous security environments and just a little work has addressed the coordination of these security modules to provide and manage End-to-End security services.

[13] presents a scalable, robust and secure distributed system that can manage communication security policies associated with multiple domains which structure can be hierarchical. It is based on a distributed system of Security Policy Servers (SPSs), each one maintaining the security policies applicable to a cluster of network entities and negotiates the security services with other SPSs. Each cluster of entities subjected to a common group of policies shall be grouped into a security domain. A collection of domains managed by a common administration will be associated with a Security Policy Server (Figure 3). The SPS determines the end-to-end path. The Policy Server receives the request from clients and other Policy Servers, processes

them and provides the appropriate policy information to the requestor based on the request and access control rules. It uses a database where the local policies and non-local policies (policies outside the boundaries of the security domain) are stored. SPP (Security Policy Protocol) is used to exchange the policy information between the clients and the Policy Servers. SPP transports policy information from the SPS Database to the security gateways and hosts.

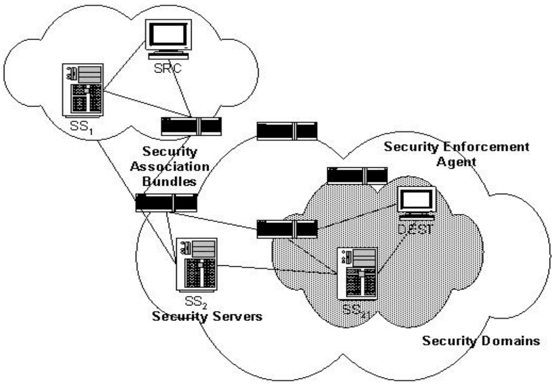


Fig. 3. Architecture of domain based security policy management system

[14] offers a good architecture called Celestial to discover the security capabilities of other domains involved in the communication path and thus it can negotiate the security context [15]. It also permits the intrusion detection and response coordination. Security Management Agent (SMA) is to be set in the management plane of any SMA-enabled node (switch, router, security gateways etc.) and is authorized to configure or reconfigure various local security mechanisms at all protocol layers. Inter-Domain SMA Coordination Protocol (ISCP) provides the transport function for the security service negotiation and reservation. The message transported by ISCP includes the security service request, security capabilities and policies of SMAs, security configuration/assignment for provisioning the requested services, and maintenance messages, etc.

The security context is established in two phases: the discovery phase where the service requirements are distributed along the communication path and the service capabilities/policies of the nodes along the path are collected; and the reservation phase which distributes assignments to the nodes selected for providing the security services. This approach offers a well-designed, reliable, scalable, secure and extensible architecture for the Inter-domain security management.

3.3 Mobility Architecture

Mobility problem based on PBNM hasn't been the subject of as many works as the QoS or Security ones. The provided solutions don't meet the requirements of an automated management of all services and users. However, some works are trying to define a model to use polices in the mobility management.

[16] uses two COPS extensions to deal with terminal and user mobility, called COPS-MU (Mobile User) and COPS-MT (Mobile Terminal) that define new policy objects in the COPS protocol for terminal and user registration. It uses also another extension called COPS-SLS (Service Level Specification) [17], combined with the former protocols to support the negotiation of QoS. Two PDPs, each representing a domain, negotiate with each other to decide the best QoS to avoid to the user/terminal.

Mobile Agents are also used to support mobility in the network. [18] uses mobile agents to provide strong mobility, which means that the service is not interrupted when the user moves from one host to another. This is accomplished by capturing the full execution state of the migrating agent. Thus, when the agent moves to another host, and having information about the previous execution state, it can continue the execution from the last state. But policy-based management is not addressed in this work.

[19] combines policies with mobile agents approach for the development of a flexible distributed system for the management of a V-Team. This work shows that it's a promising approach when combining the two paradigms (policies and mobile agents). Policies are used to monitor the behavior of an agent and its decision-making. So a policy prohibits or permits an agent to perform actions on target entities. The next section will give an overview of the mobile agent model.

4 Mobile Agent Architectures

Mobile agents (MAs) have proven their efficiency in the management of large-scale distributed and real time systems. MAs are program instances that are capable of moving within the network under their own control. They include state information (data state, execution state) that is transported within the mobile agent. They offer also a number of advantages: the saving of network bandwidth and the increasing of the overall performance by allowing the application to process data on or near the source of data, reducing network traffic, asynchronous processing (i.e. the possibility to fulfill a task without the need to have a permanent connection from the client to a host), achieving true parallel computation by employing a number of agents working on different nodes, robustness and fault tolerance [20]. Many platforms are now available and make it easy to develop a full mobile agent system. Standardization effort is also made by the OMG (Object Management Group) for a single mobile code system [21]. MAs still have some security problems like denial-of-service attacks and agent integrity. Another problem is the ease of agents' building. These issues are addressed by many agents' platforms that come with enhanced security features and facility to create and deploy mobile agent architecture. Many agent systems exist and a list of them can be found in [22].

In the context of QoS, [23] proposed an agent-based approach for QoS negotiation for multimedia applications. At each network node there is an *Agency* where fixed agents exist and where mobile agents are received, processed and dispatched. Agents interact by processing contracts between agencies via mobile agents. The C*ontract Negotiator* agents negotiate a contract on behalf of an application during session establishment. *Contract Monitor* agents are instantiated after the contract negotiators. They travel between agencies to check if the parameters stated in the contract are respected. The information about the QoS parameters for the current sessions is collected from an agent in the agency that monitors the parameters of the contract. If degradation is observed, the contract is re-negotiated or terminated. For the inter-agent communication, ORBs (Object Request Brokers) are used, and a CORBA server for the fixed agent in the agencies is implemented.

Other work on negotiation consists of the establishment of SLAs (Service Level Agreements) with policy based mobile agents [24]. It uses mobile agents to establish SLAs between ISPs (Internet Service Providers) or between an ISP and a customer in a dynamic manner. When a customer asks his ISP for a service spanning different ISPs, the customer's provider launches a negotiation process with neighboring providers using mobile agent to offer the client the requested QoS at the best cost. This approach is based on policy management architecture with a PDP (Policy Decision Point) responsible for the management of the SLAs with customers and the mobile and static agents in the domain [25]. PEPs (policy Enforcement Points) represent routers and the execution environment of the agents in the domain. This approach has the advantage of offering a dynamic establishment of an SLA and an end-to-end communication with the appropriate and common parameters.

There are many works done in the area of mobile agents and its application to networks. The management of Quality of Service in IP networks [26], which uses MAs to configure, monitor and eventually reconfigure the network, inform the client about the quality of the requested service, and permit to the administrator to manage their resources. This helps decentralize configuration and monitoring task. The discovery of and access to services in nomadic environments proposed in [27] uses a general-purpose middleware platform to enable mobile users to discover and access services efficiently in an environment subject to deficiencies and limitations since mobile devices have limited capacities and the wireless network constrained bandwidth.

5 Agent-Based Management Architecture

Since the need of a real time service provisioning, wireless networks and mobile users are growing over the Internet, using mobile agents offers better performance. Policy-based management represents also a good means of automated control. In this paper, we propose a general system for managing users in intra and inter-domain, providing their applications and services with the appropriate QoS and Security, and also managing their mobility. This architecture is based on the use of both mobile agents and policies controlling the behavior of the system (mobile agents, resources and users). Figure 4 presents the main components of this system. It is composed of:

– ***Manager Agent***: the MnA *(Manager Agent)* has the function of creating, deleting, and adding agents in the execution environment. It uses policies to manage the agents' behavior and to react to events that will happen during the run-time, like an agent's task that can't be executed, or a what to do when a foreign user connects into the domain.

- ***Coordinator Agent*:** coordinates the interaction and the information interchange between agents.
- ***Resource Agent*:** to check the available resources in the network, the MnA creates the *RA* (*Resource Agent*). RA informs the MnA for the bandwidth available and the state of the entire devices in the domain. It collects information about QoS technologies and resources of the network elements and stores them in the *Devices' Capabilities Repository*.
- ***Authentication Agent:*** the role of the AA (*Authentication Agent*) is the control of the users' identity. The user gives authentication information (login, password, address of the home domain) when connecting to the visited domain. If the address of his home domain is different from the local one, the information is transmitted by the AA to the correspondent domain (Figure 5). This means that there is a foreign mobile user whose profile must be got from the home domain. Otherwise, the AA consults the *Users' Profiles & Services Repository* to find out the user's profile. If a foreign authentication request is received, the local AA checks the local repository and sends back the corresponding profile.
- ***Policy Agent:*** the PA (*Policy Agent*) helps the domain administrator to add, edit and remove policies. It checks also the consistency of the *Policy Repository* (PR) and possible conflict between policies existing and added in the PR. The PA also provides the policies requested by the MnA if a new event occurs.
- ***Security Agent:*** the ScA *(Security Agent)* collects security information and capabilities of the existing and added devices. This information is used after by the *Negotiator Agent* (*NA*) in the negotiation process, as it knows the security technologies that can be supported in its domain. The result of the ScA is sent back to the MnA and stored in the *Devices' Capabilities Repository*.
- ***Service Agent:*** the SvA *(Service Agent)* gives the user the available services in the domain. If the user asks for a service not existing, the SvA sends a request to the neighboring domains or to the home domain to check if one of them can provide the service. If another domain can supply the service, a negotiation process takes place to decide in which condition this will be done. This is the role of the *Negotiator Agent*.
- ***Negotiator Agent:*** if a domain can't provide the requirements of a service that a user asks for, a *Negotiator Agent (*NA) is instantiated to find out common parameters between this domain and the user. The NA makes a proposal to the user's agent. This latter accepts it or makes a counterproposal. This cycle is repeated until an agreement is reached. When a contract is settled, it is sent to the MnA and stored in the *Users' Profiles & Services Repository*. The negotiation is also done when a service is localized in a neighboring domain. In this case, the negotiation includes the domain where the service is located, the current domain of the user and the user.
- ***Monitor Agent:*** has a monitoring role. It gets information from the network to make sure that the contract negotiated with the user is respected. If degradation is observed or if the parameters change from the ones set up in the contract the *Monitor Agent* (MoA) sends a report to the MnA.
- ***Billing & Accounting Agent:*** the role of the BAA (Billing & Accounting Agent) is to account for a service with defined parameters for a user. This agent also account

for a service for another domain. This is needed when a service exists only in the neighboring domains.

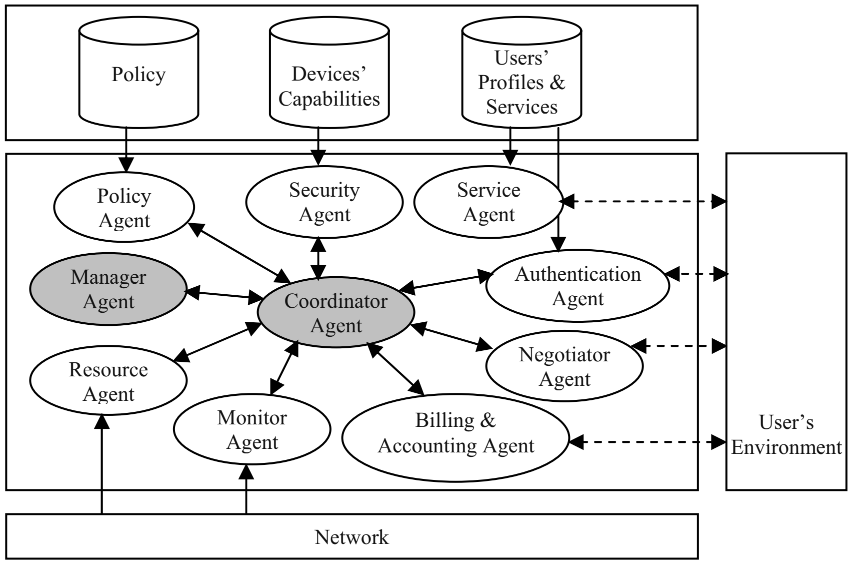


Fig. 4. Agent-based domain

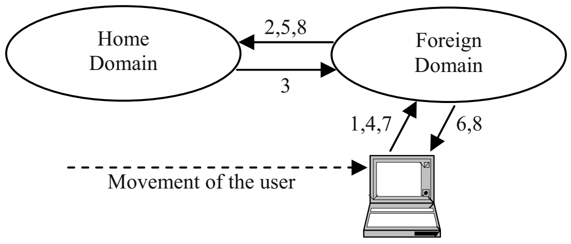
Three repositories store information:

- ***Policy Repository:*** PR stores the policies of the domain. They can be policies about what to do when a local user asks for a service, when a foreign user joins the domain, or when an agent event appears (like an agent's task that can't be done).
- ***Devices' Capabilities Repository:*** all the devices' capabilities and technologies supported are stored in the *Devices' Capabilities Repository* (DCR). The MnA uses this information to negotiate a security service or a QoS. They can also be used to map high-level policies into network level ones, since the manager knows about network capabilities, and can choose the correspondent technology to use. For example, if we have the policy:

  "*IF user1=User1 AND user2=User2 AND application = vpn THEN security= high*"

  and if we know about the security that can be implemented by a security device, we can then map the word "high" into the correspondent technology supported by this device.
- ***Users' Profiles & Services Repository:*** this repository depicts all users that have a contract with the domain and saves their profiles. It also stores, temporarily, the contracts negotiated with foreign users. The services provided by the domain are also stored with their requirements and costs. If a user asks for a service not existing in this repository (in this domain), the *Service Agent* sends a request to the neighboring domains to locate this service.

This architecture represents a service provisioning and users management system in both intra and inter-domain. The difference between the two features is if the user asking for a service belongs or not to the domain and if the requested service spans multiple domains. If the user doesn't belong to the domain, the conditions under which the service will be provided must be defined by negotiating the service's parameters. The steps needed are given by figure 5.



1: Authentication request
2: Authentication request
3: Profile sending
4: Service request
5: Information about the service requested by the mobile user
6: Proposal
7: Agreement
8: Service established with the negotiated parameters

Fig. 5. Different steps for a negotiation process with a mobile user

6 A Negotiation Scenario

In this section, we give an example scenario of an inter-domain negotiation where a mobile user "User1" whose home domain is "Domain1", moves to a foreign domain "Domain2". We give a description of the steps done from the connection of the user into "Domain2" until the provision of the requested service (figure 6).

When in "Domain2", an *Authentication Agent* (AA) gets the necessary information that authenticates the user from the user's environment. Since the address of the domain is "Domain1", AA moves then to the home domain of "User1" where it gets the user's profile from the home AA. To provide the user's profile, the home AA consults the home *Users' Profiles & Services Repository* and search for the corresponding entry to "User1". The AA moves back to "Domain2" and stores the user's profile in the *Users' Profiles & Services Repository* of "Domain2". Once authenticated, "User1" sends a request asking for a service. Say "User1" asks for a videoconferencing service under a VPN (Virtual Private Network) channel with a "CN" (Correspondent Node) in "Domain1". SvA checks if the videoconferencing and VPN services can be provided in "Domain2" by consulting the *Users' Profiles & Services Repository*. SvA moves also to "Domain1", to inform it about the services requested. If the services can't be provided in the visited domain, SvA informs "User1" about this. Otherwise, it transmits the information to the Manager Agent (MnA). The MnA initiates a Policy Agent (PA) to get the necessary policies from the policy repository. The MnA also launches a Resource Agent (RA) to get information about the available resources in the network. If the policies restrict the QoS and Security offered to "User1", or if the requirements of the services can't be satisfied due to a lack of resources, the *Negotiator Agent* negotiates the QoS and security parameters with "User1".

If this process ends with an agreement, the result is sent back to the home domain informing it about the parameters that will be set. The services are then implemented and the user informed. The *Monitor Agent* (MoA) checks if the parameters negotiated are respected. If one of them exceeds a threshold, or if a problem occurs during the service, MoA informs the MnA. The threshold is set in the negotiation phase defining the value under which the service is considered degraded.

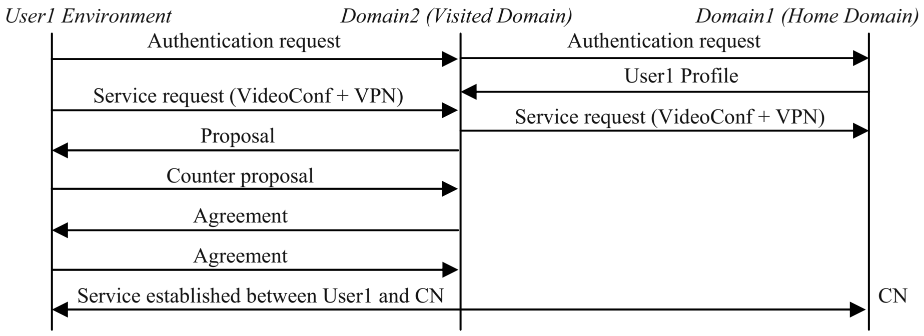


Fig. 6. A negotiation process

If the service spans multiple domains, the *Service Agent* is sent to each domain informing it about the services required. The *Negotiator Agent* moves also to each domain to conduct the negotiation. At the end, each agent moves back to the foreign domain. There is no negotiation with the home domain since "User1" has already a contract with it. If the negotiated parameters are not the same for all the domains, MnA chooses the fewer ones.

7 Conclusion and Future Work

This paper presents work done in the area of network management of QoS, Security, and mobility. Mobile agent architectures were also given to show how they could help to resolve many problems encountered in the limitation of network resources and the real-time service provisioning. Each work defines how to establish and manage a communication, and how to monitor it. Different management policies exist to supply a communication session with the parameters required, resulted (after negotiation) or contracted for (with SLA). But it's still hard to deal with all the issues in a single architecture.

We focus on defining a management system that manages the establishment of communication sessions and administers the users and their mobility to supply them with the needed services wherever they would be. The architecture proposed defined the entities required to manage users in their home domains and in foreign ones and to set up the appropriate requirements (QoS and security needs) of the service even if it spans multiple domains. Mobile agents are a good alternative since we can have wireless mobile users, or wireless networks. They aren't sensible to the cut of the connection and offer a good bandwidth saving. Thus, a wireless user sends a proposal with a *Negotiator Agent* and then disconnects. After he connects again, the mobile agent moves to its home environment and informs the user about the parameters negotiated. Policies manage the behavior of the entire system. They control the life cycle of the agents and what to do if one of them has a problem during its run time, or what to do when a foreign user joins the domain and which resources are offered to him.

The point that must be addressed is to specify policies to guarantee that the service established would not be deteriorated and also what to do when there is a dysfunction in a network entity. We have to apply this architecture using an existing platform. We explore the use of an agent platform offering an easy deployment of agents, and a good security level for agents and execution environments. We can use one of the agent platform listed in [22]. The modeling of the policies could be done using one of the policy languages presented in section 2. We are also investigating the mapping of policies from high level to network level ones. To do so, we can use the information provided by the security and resource agents, since they give information about the entities in the network and the technologies supported by each one. We are elaborating an approach to model the devices' capabilities and to find a means for gathering the devices information. An evaluation of the performance will quantify the delay of the provision of a service spanning two or more domains, for users from the authentication to the establishment of this service.

References

1. N. Dulay, E. Lupu, M. Sloman, and N. Daminanou. "A policy Deployment Model for the Ponder Language". In Proc. of the 7th IFIP/IEEE International Symposium on Integrated Network Management (IM'2001), Seattle, Washington, USA.
2. S. Jajodia, P. Samarati, and V. S. Subrahmanian. "A Logical Language for Expressing Authorizations". In Proc. of IEEE Symposium on Security and Privacy, 1997, pp.31-42.
3. R. Ortalo. "A Flexible Method for Information System Security Policy Specification". In Proc. of the 5th European Symposium on Research in Computer Security (ESORICS 98), 1998, Louvain-la-Neuve, Belgium.
4. J. A. Hoagland, R. Pandey, and K.N. Levitt. "Security Policy Specification Using a Graphical Approach". Technical report CSE-98-3, UC Davis Computer Science Department, July 22, 1998.
5. J. Lobo, R. Bhatia, and S. Naqvi. "A Policy Description Language". In Proc. of AAAI, July 1999, Orlando, Florida, USA.
6. N. Damianou, N. Dulay, E. Lupu, and M. Sloman. "The Ponder Specification Language". In Proc. of Policy 2001, Workshop on Policies for Distributed Systems and Networks, Bristol, UK, 29-31 Jan. 2001, Springer-Verlag LNCS 1995, pp. 18-39.
7. C. Goh. "Policy Management Requirements". System Management Department, HP Laboratories, Bristol, technical report HPL-98-64, April 1998.
8. Hewlett-Packard Company. "A Primer on Policy-Based Network Management". OpenView Network Management Division, Hewlett-Packard Company, September 24, 1999.
9. S. Duflos, G. Diaz, V. Gay, and E. Horlait. "A Comparative Study of Policy Specification Languages for Secure Distributed Applications". DSOM 2002 IFIP/IEEE, LNCS 2506 Springer-Verlag, pp157-168, Montreal, Canada, October 2002.
10. N. C. Damianou. "A Policy Framework for Management of Distributed Systems". PhD thesis, Imperial College of Science, Technology and Medicine, February 2002.
11. V. Sander, W.A. Adamson, I. Foster, and A. Roy. "End-To-End Provision of Policy Information for Network QoS". In Proc. of the 10th IEEE Symposium on High Performance Distributed Computing (HPDC), IEEE press, August 2001.

12. T. Ebata, M. Takihiro, S. Miyake, M. Koizumi, F. Hartanto, and G. Carle. "Interdomain QoS Provisioning and Accounting". Internet-Draft, draft-ebata-interdomain-qos-acct-00.txt, November 1999.
13. J. Zao, L. Sanchez, M. Condell, C. Lynn, M. Fredette, P. Helinek, P. Krishnan, A. Jackson, D. Mankins, M. Shepard, and S. Kent. "Domain Based Internet Security Policy Management ". Proceedings of DARPA Information Survivability Conference and EXposition 2000 (DISCEX '00), Jan 25-27, 2000.
14. F. Gong, Y. F. Jou, C. Sargor, and S. F. Wu. "Architecture Design of the Celestial Security Management System". Jan. 28, 1998.
15. Z. Fu, H. Huang, T. Wu, S.F. Wu, F. Gong, C. Xu, and I. Baldine. "ISCP: Design and Implementation of an Inter-Domain Security Management Agent (SMA) Coordination Protocol". In Proc. of NOMS 2000, April 2000, pp 565-578.
16. G. Pujolle, and H. Chaouchi. "A New Policy Based Management of Mobile Users". Networking 2002, LNCS 2345, pp. 1099-1104, 2002.
17. G. Pujolle, and H. Chaouchi. "QoS, Security, and Mobility Management for Fixed and Wireless Networks under Policy-Based Techniques". IFIP World Computer Congress, 17th edition, August 25-30, 2002, Montréal, Canada.
18. N. Suri, J. M. Bradshaw, M. R. Breedy, P. T. Groth, G. A. Hill, and R. Jeffers, "Strong Mobility and Fine-Grained Resource Control in NOMADS". ASA/MA 2000, Zurich, Switzerland.
19. H. Harroud, M. Lakhdissi, A. Karmouch, and C. Grossner. "Policy-Based Management for Multimedia Collaborative Services". MMNS 2001, LNCS 2216, pp. 285-298, 2001.
20. A. Bieszczad, B. Pagurek, and T. White. "Mobile Agent for Network Mangement", IEEE Communication Survey, Fourth Quarter 1998, Vol 1, No. 1. http://www.comsoc.org/pubs/surveys.
21. D. Milojicic, M. Breugst, I. Busse, J. Campbell, S. Covaci, B. Friedman, K. Kosaka, D. Lange, K. Ono, M. Oshima, and C. Tham. "MASIF: The OMG Mobile Agent System Interoperability Facility", available on: http://www.hpl.hp.com/personal/Dejan_Milojicic/ma4.pdf.
22. http://mole.informatik.uni-stuttgart.de/mal/mal.html
23. L. A. Guedes, E. Cardozo, and P. C. Oliveira. "An Agent Based Approach for Quality of Service Negotiation and Management in Distributed Multimedia Systems". First International Workshop on Mobile Agents 97 (MA' 97), Berlin, Germany, April 7-8, 1997.
24. M. Fonseca, and N. Agoulmine. "Policy Based Agents as a Solution for Inter carriers SLA Negotiation to support End-to End IP QoS". IEEE GLOBECOM 2002, Taipei, November 2002.
25. N. Agoulmine, M. Fonseca, and A. Marshall. "Multi-domain Policy Based Management Using Mobile Agents". Mobile Agents for Telecommunication Applications, Third International Workshop, MATA 2001, Montréal, Canada, August 14-16, 2001.
26. T. Mota, S. Gouveris, G. Pavlou, A. Michalas, and J. Psoroulas. "Quality of Service Management in IP Networks Using Mobile Agent Technology". Mobile Agents for Telecommunication Applications, Fourth International Workshop, MATA 2002, pp.193-205, Barcelona, Spain, October 2002.
27. Z. Wang, and J. Seitz. "Mobile Agents for Discovering and Accessing Services in Nomadic Environments". Mobile Agents for Telecommunication Applications, Fourth International Workshop, MATA 2002, pp. 269-279, Barcelona, Spain, October 2002.

Context-Aware Services Provisioning on Top of Active Technologies

Irene Sygkouna, Stavros Vrontis, Maria Chantzara,
Miltiadis Anagnostou, and Efstathios Sykas

National Technical University of Athens, 9, Heroon Polytechneiou Str., 15773, Zografou, Athens, Greece
{isygk,marhantz,stvront,miltos,sykas}@telecom.ntua.gr

Abstract. With respect to the pervasive computing vision, the paper deals with the efficient provisioning of context-aware services making use of Active Networks (ANs) on top of fixed and mobile infrastructure. This represents a starting point for studying on the utilization of Active Technologies in order to provide context-aware services. In these terms, Mobile Agent (MA) Technology comprises a type of Active Technology and the exploitation of its utilization in the field of context-awareness is analyzed. Following, an evaluation comparison of ANs and MAs paradigms is given, identifying the drawbacks and the advantages of each paradigm. Finally, some concluding remarks regarding the combined usage of them are presented.

1 Introduction

Weiser's [1] "ubiquitous computing", which represented the vision of people and environments augmented with computational resources that provide information and services when and where desired, has excited many researchers. Stemming from this notion, context-aware computing demonstrates promise for making our interactions with services more seamless and less distractive from our everyday activities. As stated in [2], a context-aware system can be seen as a human assistant given user's context to be responsible to make decisions in a proactive fashion, anticipating user needs while not disturbing the user, except for an emergency. The key point is how to better construct a context-aware system that could emulate such a human assistant.

On the other hand, the need for development of more complicated services that react with the network infrastructure has guided the researchers to transfer efficiency into the network nodes. Following this trend, the term "Active Technologies" has emerged, including paradigms that support computational models for the utilization of distributed computing resources inside the network. In this perspective, the IST project CONTEXT [3] is elaborating on the efficient provisioning of context-aware services making use of ANs on top of fixed and mobile infrastructure.

CONTEXT represents a starting point for studying on the utilization of other Active Technologies in order to provide context-aware services. In these terms, MAs paradigm comprises a type of Active Technology and the exploitation of its utilization in the field of context-awareness can prove to be beneficial.

This paper aims to examine the usage of Active Technologies for providing context-aware services. The explored technologies are ANs and MAs. The paper is organized as follows: Section 2 tackles the issue of context-aware service provisioning in terms of the classification and modeling of context information (section 2.1) and CONTEXT architecture for service provisioning (section 2.2). In Section 3, after identifying the benefits gained from active technologies, the implementation aspects of ANs (section 3.1) and MAs (section 3.2) are analyzed. Section 4 presents an evaluation of these active technologies. Finally, Section 5 provides some conclusive remarks and future plans regarding the presented research.

2 Context-Aware Services Provisioning

As computer and network become more pervasive, the nature of services should change accordingly. Services must become more flexible in order to respond to highly dynamic computing environments, and become more autonomous to satisfy the growing and changing requirements from users. Therefore, the need for context -aware services becomes imperative. CONTEXT will provide context -aware services based on the following procedure: specifying what context an application needs, gathering the required context and performing the appropriate actions in order to ensure transparent adaptability to the changeable environment. Following, we deal with classification and modeling of context information in order to introduce an architectural view of our context -aware services provisioning system.

2.1 Classification and Modeling of Context Information

Context can be quite rich, consisting of any attributes that would make a service function in a more proactive manner reflecting user's needs. A classification according to the nature of context information identifies the following context categories: 1) User Info, e.g. name, age occupation, accounting info, calendar/agenda info, 2) Personal Info, e.g. health, mood, hobbies, interests, 3) Activity Info, e.g. "what the user is doing now", "with whom is he now", 4) Social Info, e.g. friends, family, employer, employee, colleagues, 5) User Defined Rules, e.g. not to be disturbed when being at place x or with person y, 6) Environment Info, e.g. location info, weather, surrounding facilities, 7) Application Info, e.g. application related issues, 8)Terminal Info, e.g. type of terminal, type of connection, installed toolkits/programs, installed peripherals, 9) Network Info, e.g. characteristics of network connections.

In order to cope with context information handling, a special structure is used, named Context Entity (CE) that consists of information attributes featuring a specific type of context information and interfaces indicating the means to retrieve this information. Each such interface corresponds to a capability provided by the underlying network and in this case we refer to capabilities that return acquired information. The actions that will be performed are modeled through another structure, named Action Entity (AE) that consists of interfaces representing the means to invoke these actions and corresponding to capabilities that apply certain configuration actions on the network infrastructure, invoke other services, or notify users.

2.2 Architecture Design of CONTEXT

CONTEXT project is currently working on the definition of a middleware that aims to offer an automatic creation, delivery and management of context-aware services. The model of context-aware services is based on policies and the Policy Core Information Model (PCIM) [4] is used. Based on the pattern that CONTEXT proposes, a service developer is enabled to construct a context-aware service using as building blocks a set of predefined CEs and AEs. The developer, through a graphical environment, just picks the available entities in order to produce a complete description of the service and based on this description, the necessary mechanisms for deploying the service are created. Following, through the subscription phase, the customer enters the attributes that customize the service and finally, through a set of responsible components, the service logic corresponding to each customer is produced (Service Logic Object - SLO) and deployed on the execution environment. The execution environment of the produced context-aware services resides within the AN platform that CONTEXT is going to utilize.

During the operation phase of a context-aware service, context information is acquired, evaluation of policies is performed and the appropriate actions to be enforced are triggered, according to the logic of the service. However, the richness of context information and the fact that context may be acquired from multiple context sources, represent the key motivation behind the introduction of a module named Context Information Access Handler (CIAH). CIAH is going to handle the gathering and dissemination of context information as well as the triggering of actions imposed by the service logic. The introduction of CIAH provides abstraction of context acquisition from applications and makes easier the development of applications. Fig.1 depicts the interactions of CIAH with the SLOs and the underlying network. As it is shown in the figure, CIA H is also deployed on the AN layer.

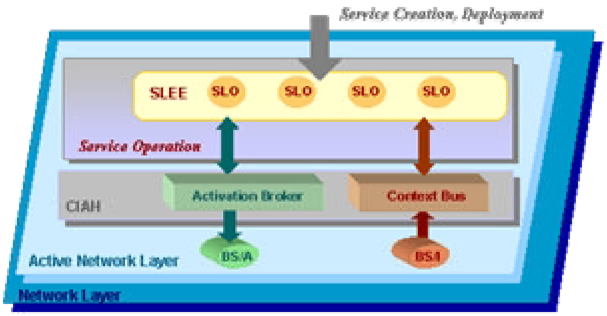


Fig. 1. CIAH Interactions

CIAH receives requests from the SLOs asking for a specific type of context information. Context Bus sub-module contacts the appropriate context information sources and disseminates the context value to the requestor SLO. Context Bus also handles SLO requests asking to be notified when a specific context value occurs. The aforementioned context retrieval mechanisms correspond to the "pull" and "push" mechanisms of receiving information. Activation Broker sub-module is responsible to trigger the actions imposed by the service logic.

In order for the CIAH to accomplish the aforementioned tasks, it interacts with the Basic Services (BSs) that are implemented on top of the AN platform and reflect the capabilities provided by the AN. BSs are offered by the active application layer and can be discriminated in those that obtain context information and are named Info BSs (BS/I), and those that are responsible to trigger actions and are named Action BSs (BS/A). BSs are distinguished from the offered context-aware services that the CONTEXT service platform is going to produce, in the sense that they are considered as auxiliary services that enable context-aware services to take advantage of the capability features offered by the AN infrastructure.

CONTEXT architecture exploits the benefits of implementing context-aware services with the use of AN technology. This architecture represents the motivation in order to study on the utilization of other Active Technologies for the provisioning of context-aware services. Following, we identify the benefits gained from Active Technologies and exploit apart from the ANs paradigm, the integration of MAs Technology in the field of context-awareness.

3 Utilization of Active Technologies

The development of context-aware services has been hampered by the need to develop a custom context infrastructure for each application. CONTEXT removes this obstacle by placing the context functionality in the network infrastructure in a form of active components. The development of middleware solutions for an efficient provisioning of context-aware services calls for distributed control and programmability inside the network. This will allow the dynamic adaptability of current and future context aware services for the benefit of the global consumer.

The need for distributed control derives from the fact that the various context sources are distributed among the network nodes. A centralized scheme for management control operations would hardly be scalable and would not allow efficient operation. On the other hand, the realization of the concept of management by delegation through the distribution of code to core network nodes will reduce the number of necessary network transactions by local processing.

Apart from distributed operation and control, programmability inside the network is another prerequisite. Building nodes that can conduct appropriate computations and thus gaining the ability to dynamically change network functionality would enable us to implement new network capabilities in a flexible way. These capabilities would be software–based allowing dynamic customization of network resources and thus, the services provided would adapt to context changes without any noticeable changes to the upper levels [5, 6].

Our approach to develop an adaptive service and context execution environment for next-generation networks, which is inherently distributed and programmable, is based on using Active Technologies on top of fixed and mobile networks. We use the term Active Technologies to include the recently developed paradigms of Active Networks (ANs), Programmable Networks (PNs), Mobile Agents (MAs) and Semantic Networks (SNs). Although these concepts were introduced by different research communities to address different problems, they have started overlapping in focus and applicability, as they are being developed further [7]. CONTEXT will make use of the Active Technology as the basic deployment, delivery and assurance mechanism that at the same time supports context information and high bandwidth requirements for future applications. The challenge such an infrastructure poses is to find the right balance among flexibility, performance, robustness and usability [6]. Following this direction, we study two different Active Technology paradigms in order to provide context-aware services: ANs and MAs.

3.1 Active Networks

ANs is a framework where network elements, primarily routers and switches, are programmable. Programs that are injected into the network are executed by the network elements to achieve higher flexibility and to present new capabilities. ANs can work in two different ways: users can preprogram selected network nodes, such as routers, with customized code before running an application, or they can program data packets, which then transport code to nodes along the way to their destination [5].

CONTEXT utilizes Active Bell-Labs Engine (ABLE) [8] system developed in Lucent Technologies to exploit ANs. ABLE is a modular and scalable software architecture that enables the deployment, control and management of active services, usually called active sessions, over network entities such as routers, WLAN access points, media gateways and servers supporting such services in IP-based networks.

The active node is composed of a Forwarding Element (FE), namely router, WLAN access point, media gateway, etc. with an inherent diverter that detects and diverts active packets to the main separate component, the Active Engine (AE), and the AE, which executes the code of the active packet (see Fig.2). The FE and the AE can be either physically separated or located at the same machine. The AE consists of the following modules: the Session Broker, which receives and parses active packets, handles and manages existing services, and distributes active packets according to requests of the services, the Info Broker which provides an efficient monitoring channel by exporting local state to active sessions and mediates all queries for local state and the Control Broker, which provides a secure channel for control and configuration operations on the active node. During the progress of CONTEXT, ABLE is expected to become extended, accommodating new modules in order to meet the forthcoming project needs. In this sense, the need for one additional module has currently been identified. This is called Context Broker and provides capabilities related to other than network-minded context, such as users, environment, application or terminal related context. The modules communicate through UDP transactions and TCP connections.

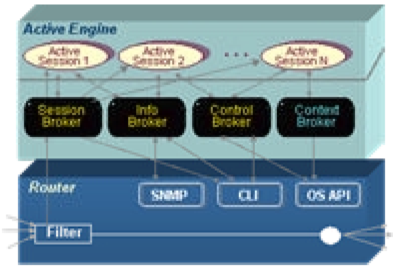


Fig. 2. ABLE Active Node Architecture

Each node can communicate with other active nodes in the following ways: the topology-blind addressing mode, which enables a node to send a packet to the nearest active node in a certain direction, removing the need for having available full topology information for any specific node, and the explicit addressing mode, which enables a node to send a packet to a specific active node [9, 10].

The testbed network infrastructure that will be used for the trial of the Context-Aware Services is illustrated in Fig.3. Specifically, it consists of two Linux-based active routers (PCs running Linux with the AE residing in the same machine with the router), two Cisco routers with adjunct active engines, and a non-active Cisco router. Apart from the IP network capabilities, Wireless LAN along with GPRS capabilities will be controlled by active nodes through the corresponding wrappers that ABLE is going to accommodate. The end-users will be able to access context-aware services using personal computers, laptops, PDAs and mobile phones.

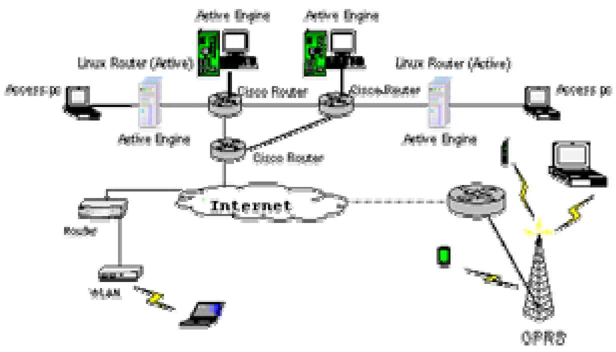


Fig. 3. Testbed Network Infrastructure

3.2 Mobile Agents

The other approach that will be exploited in the framework of the Active Technologies applicability is the MAs paradigm. The MA concept is a computational paradigm characterized by computer programs, called agents, migrating inside the network and executing on network nodes. In this sense, a MA that moves from one node to another can be interpreted as an active packet containing data and a program, which is sent from one node to another in an AN [7].

Mobile agents can move around and execute tasks autonomously and in line with their goals which can dynamically change according to the changeable nature of context information. They can also be intelligent, and this makes them even more interesting for Active Technologies. An intelligent piece of code that moves around can be considered the most advanced form of active code. All other forms derive from this one by combining some but not all characteristics mentioned within the intelligent and mobile agent research domain [11].

The architecture of the active node that will be able to host and execute the active code is depicted in Fig.4. On the bottom, there is the hardware part of the node. Then, the operating system, Linux, will export an open interface that represents the abstraction of hardware resources. The Execution Environment is provided by the MA platform, Grasshopper [12]. This is the agency as described in Mobile Agent System Interoperability Facility (MASIF) [13] standard. Each agency corresponds to a different active session and consists of places. A place is a context within an agency in which an agent is executed. The place can provide various functions such as access to local resources.

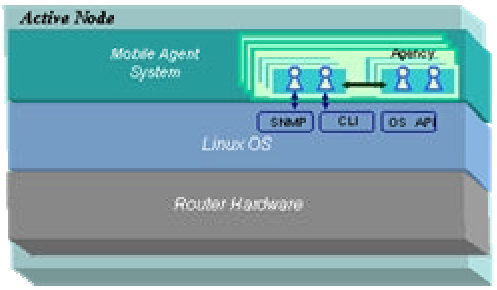


Fig. 4. Active Node for MAs

Mobile and stationary agents cooperate in an agency in order to provide programmability to the node. The Info Stationary Agent (SA/I) access router information through Simple Network Management Protocol (SNMP) interface [14] while the Action Stationary Agent (SA/A) apply configuration actions on the nodes through the Command Line Interface (CLI) or other Operating System interfaces (OS API). Consequently, SA/I along with SA/A agents provide the interfaces required for a mobile agent to access node's hardware and internal software functions. Each mobile agent implements a different BS, as has already been named a certain capability of the Active Application Layer. The mobile agents can be sent dynamically “on-demand” to routers where they are currently required. After their arrival, they interact with the appropriate stationary agents in order to fetch the required context information or apply the necessary configuration actions. For enabling mobile agents to communicate with different controlled elements, such as Linux routers, Cisco routers, WLAN access points or GPRS gateways, the stationary agents provide the corresponding wrapper functionalities.

4 Evaluation of Active Technologies

Although the motivations for proposing the MAs and ANs paradigms were quite different, coming from different research communities, these two technologies nowadays tend to share common ideas. They both overcome the traditional client/server network management model achieving better scalability through distribution and more efficient usage of resources.

However, these paradigms have some crucial differences. First of all, they are implemented at different layers of the OSI model. The MAs paradigm runs at the application layer while the ANs one at the network layer. Consequently, in the case of ANs no special treatment has been taken into account for the migration mechanism of the active code since the network elements contain sophisticated software to perform packet forwarding, and thus, the main concern is the active processing of the packets in the network infrastructure. Working at the network layer reveals the real potentiality of ANs to exploit network services more efficiently and perform fast deployment of distributed applications. On the other hand, the lowest layer of the OSI model that MAs consider programming for is the transport layer, simply making use of the communication capabilities of the environment (e.g., Java RMI). In this sense, working at the application layer, makes the MA paradigm outperform when application tasks are performed. This can be particularly interesting in the case of context-aware services since the retrieval of context information along with the actions applied do not merely imply interaction with network infrastructure but also with users, external systems or applications. This means that not only management of routers is required but also management of network servers that host applications providing context information, and in the latter case the MA paradigm is supposed to perform more efficiently [15, 16].

Secondly, as also mentioned above, the ANs paradigm supports two communication modes: the topology-blind addressing mode, in which an active packet is executed on every active node in a certain direction, and the explicit one, in which a packet is executed only on the target active node, while it is just forwarded by the intermediate ones. On the contrary, an agent can only be sent explicitly to a specific node in order to be executed. The topology-blind addressing mode of ANs offers dynamic and thorough customization of network resources even if the application does not have knowledge of the current underlying network infrastructure. However, there are applications that require the actions to be taken on specific nodes that are known a priori. In such cases, the MAs paradigm outperforms since it avoids executing the active code on every active node and eliminates the additional delay for diverting an active packet from the router to the AE, which appears in ANs.

Thirdly, the operation of ANs requires advanced router configuration in terms of special policies that enable the router to identify the active packets and treat them accordingly, namely forward them to the appropriate AE in order to be executed. In this sense, upgrading a node to an active one is more difficult in case of ANs than in MAs.

Last but not least, it is worth mentioning that while the payload of an active packet in case of ANs consists only of the program code and the data, a mobile agent transfers additionally its state while it traverses the network. On the grounds that an agent can make decisions based on this state, it migrates in the network under its own control, namely it can stop its execution in one node and continue in another with respect to the volatile nature of context information [17].

5 Conclusions and Future Work

The fact that context may be acquired from multiple distributed and heterogeneous sources as well as the fact that it changes dynamically make it a really hard work to build context-aware services. Our approach to develop a middleware for efficient provisioning of context-aware services is based on using Active Technologies on top of fixed and mobile networks. The exploitation of ANs and MAs paradigms in the field of context-awareness has been examined, highlighting the different prospects of each one.

The motivation behind ANs comes from the emerging concern on how to best adapt the existing networks in order to accommodate new type of applications such as context-aware ones. On the other hand, MAs paradigm pays less attention to the communication model that is ultimately based on high-level abstractions, exploiting it to implement distributed applications. As a consequence, a combined approach that exploits both the ANs and MAs paradigms is considered to be most fruitful based on the concept that MAs paradigm could be inspired by the ANs communication model in order to access network resources more efficiently. Alternatively, each context-aware application could use for some tasks the one paradigm and for some others the other, depending on the estimated efficiency achieved in each case. A performance evaluation both from an application and network point of view will be regarded, in terms of time and communication complexity respectively. Finally, the idea of exploiting special navigation patterns for coping with navigation and synchronization aspects of an active process will be taken into account, as it is considered essential for performance improvement.

Being inspired by both technologies, in the long term, it may be appropriate to merge these two paradigms, to form an even stronger vision of how we can exploit the potential of networks to the fullest [15].

References

1. M.Weiser, "The Computer of the 21st Century", Scientific American, Vol. 265, No. 3, pp.66-75, September 1991

2. M. Satyanarayanan, "Challenges in Implementing a Context-Aware System", editorial introduction in IEEE Pervasive Computing, pp.2, July-September 2002
3. CONTEXT: Active Creation, Delivery and Management of efficient Context Aware Services, IST-2001-38142-CONTEXT, http://context.upc.es
4. IETF Policy Core Information Model Extensions, http://www.ietf.org/internet-drafts/draft-ietf-policy-pcim-ext-08.txt, May 2002
5. Sixto Ortiz Jr., "Active Networks: The Programmable Pipeline", IEEE Computer Magazine, Vol.31, No.8, pp.19-21, August 1998
6. S. Karnouskos, "Realization of a secure active and programmable network infrastructure via mobile agent technology", Computer Communications, Vol. 25, Issue 16, pp.1465-1476, October 2002
7. R. Kawamura, R. Stadler, "Active Distributed Management for IP Networks", IEEE Communications Magazine, pp.114-120, April 2000
8. ABLE: "Active Bell Labs Engine", http://www.cs.bell-labs.com/who/ABLE
9. D. Razz, Y. Shavitt, "An Active Network Approach for Efficient Network Management", IWAN '99, July 1999, Berlin, Germany
10. D. Razz, Y. Shavitt, "Active Networks for Efficient Distributed Network Management", IEEE Communications Magazine, 38(3), pp.138-143, March 2000
11. H.S. Nwana, D.T. Ndumu, "A brief introduction to software agent technology", N. Jennings, M. Wooldridge (Eds.), Agent Technology Foundations: Applications and Markets, Springer, Berlin, 1998
12. IKV++ Technologies AG (2002), Grasshopper MAP, http://www.grasshopper.de
13. OMG MASIF Standard, http://www.fokus.gmd.de/research/cc/ecco/masif/
14. SNMP research, http://www.snmp.org
15. M. Collier, "Mobile Agents and Active Networks: Complementary or Competing Technologies?", Technical Report of the Broadband Switching & Systems Laboratory
16. Ichiro Satoh, "A Mobile Agent-based Framework for Active Networks", in Proceedings of IEEE Systems, Man, Cybernetics Conference (SMC'99), pp.71-76, IEEE, October 1999
17. H. Liu, M. Nodine, "On Applying Mobile Agents to Network Management", in Proceedings of INET '96, Montreal, June 1996

COSMOS: A Context-Centric Access Control Middleware for Mobile Environments

Paolo Bellavista, Rebecca Montanari, and Daniela Tibaldi

Dipartimento di Elettronica, Informatica e Sistemistica Università di Bologna Viale Risorgimento, 2 – 40136 Bologna – Italy {pbellavista, rmontanari, dtibaldi}@deis.unibo.it

Abstract. User/terminal mobility during service provisioning and high heterogeneity of wireless portable devices identify novel challenges for service delivery in ubiquitous pervasive environments. An emerging architecture solution in the wireless Internet is to have middleware components (mobile proxies) over the fixed network that follow the movements and act on behalf of the limited wireless clients. It is crucial that mobile proxies have full visibility of their context, i.e., the set of available and relevant resources depending on access control rules, client location, user preferences, privacy requirements, terminal characteristics, and current state of hosting environments. The paper presents the design and implementation of a context-centric access control middleware, called COSMOS, for the wireless Internet. COSMOS dynamically determines the contexts of mobile proxies, and effectively rules the access to them, by taking into account different types of metadata (user profiles and system/user-level authorization policies), expressed at a high level of abstraction and cleanly separated from the service logic. The paper also shows how COSMOS facilitates the development of articulated access control strategies in the case study of a context-dependent movie-info service deployed over IEEE 802.11 network localities.

1 Introduction

Telecommunication systems and the Internet are converging towards an integrated pervasive scenario that permits the development and the access to services in highly dynamic mobility environments. Mobile users can connect to the Internet from ubiquitous access points, possibly by preserving their personal preferences and the state of their ongoing sessions. Mobile devices can either nomadically connect/disconnect to/from different points of attachment or continuously maintain connectivity during their roaming [1]. In particular, the recent proliferation of heterogeneous portable devices and of different technologies for wireless connectivity in home/office environments suggests not only extending to mobile users/devices the access to traditional Internet services, designed and implemented for the fixed network infrastructure, but also to develop new classes of services that can provide results depending on the relative position of clients and on the consequent resource visibility. For instance, a city guide assistance service should dynamically retrieve maps, audio descriptions, and detailed textual information about the buildings, restaurants and theatres close to the current user location. Service results, possibly retrieved from traditional Web servers, should be filtered and downscaled at provisioning time depending on the hard-ware/ software limitations of client access devices.

The complexity of the above scenario calls for novel middleware solutions for facilitating service development and for supporting service delivery to wireless devices in mobility environments. We claim that these novel middlewares should exhibit the non-traditional property of context-awareness. Different definitions of context have been recently proposed: for example, Schilit and Theimer identify context as the identities and locations of nearby users and objects, and their modifications during service sessions [2], while Pascoe uses the term context to indicate the subset of physical and logical states of interest for a specific subject [3]. In the following, we adopt a large definition of context as the collection of any information useful to characterize the runtime situation of a user during her service session [4]. In addition, the middleware should enable the dynamic installation/migration of client-side middleware/service components and their seamless discarding when no longer needed.

By focusing on a commonly addressed networking scenario, in the following we will use the term wireless Internet to indicate the integrated networks where wireless solutions extend the accessibility of the fixed Internet via service access points working as bridges between fixed hosts and wireless devices, such as in the case of IEEE 802.11 access points providing Wi-Fi equipped laptops with connectivity to a traditionally wired local area network.

Few research activities have recently started to investigate middleware solutions for the wireless Internet: an emerging guideline is to have active middleware components that are deployed at service provision time to act as user/device proxies over the fixed network and to carry on the needed service configuration, tailoring and management. The relevance of implementing middleware components as mobile proxies that follow the client movements and execute in their same network locality starts to be recognized [5, 6]. We claim the suitability of designing and implementing mobile proxies for the wireless Internet in terms of Mobile Agents (MAs). MA-based proxies can carry on service requests autonomously even in case of temporary device disconnection, can migrate among different network localities by maintaining the session state, and can exploit the full context visibility typical of the MA programming paradigm to support context-based service configuration and tailoring [4].

However, the deployment of MA-based proxies also raises novel and challenging security concerns [7, 8, 9]. On the one hand, the possible injection of malicious proxies can compromise the security of the wireless Internet fixed nodes similarly to the case of viruses and worms. On the other hand, malicious nodes may try to disclose the private information carried by the hosted proxies and to tamper with the MA code and state. In addition, adopting a middleware solution based on mobile proxies requires establishing a secure user-proxy trust relationship to ensure that proxies actually operate according to the user requirements [10].

The paper focuses on the novel access control issues coming from the adoption of MA-based proxies for the support of context-dependent service provisioning over the wireless Internet. Several practical techniques have been already proposed to control and confine the interactions between MAs and hosting execution environments. Type-safe languages permit to determine whether incoming MAs respect safety properties, such as address space confinement. Sandboxing techniques can be used to rig-idly limit the resource visibility and access scope of MAs while executing and have evolved to propose more advanced access control solutions [11]. However, these solutions are not flexible enough for the addressed highly dynamic pervasive environments. Frequent and unexpected proxy mobility, to follow the recurrent movements of wireless devices, significantly increases the complexity of access control decisions. Hosting environments interact with MA-based proxies they are not familiar with and, most important, whose identity may be un-informative or not sufficiently trustworthy. It is unlikely for hosting environments to know in advance the identities/roles of all subjects that will request the access to their managed resources/services, thus making inappropriate traditional access control policies based on subject identity/role. In other words, we claim that context-dependent service provisioning in the wireless Internet requires a deep paradigm shift from subject-centric access control models to context-centric ones: access control policies should also depend on the dynamic evaluation of the applicable context.

The paper presents the design and implementation of a highly dynamic and flexible security middleware, called COSMOS (**CO**ntext-aware **S**ecurity Middleware for **MO**bile Agent **S**ystems), for context-centric access control in the wireless Internet. COSMOS dynamically determines the contexts of MA-based proxies, and effectively rules the access to them, by taking into account different types of metadata (user/device/resource profiles and system/user-level authorization policies), expressed at a high level of abstraction and cleanly separated from the service logic. COSMOS provides an integrated environment for both the specification of context-dependent access control policies and for their runtime enforcement. The proposed access control middleware integrates with the SOMA framework which offers the support facilities for user/terminal mobility and for the proxy-based discovery/binding of wireless devices to the needed resources in their contexts [4, 12].

The paper finally presents the case study of a context-dependent movie-info service prototype, built on top of COSMOS and deployed over a group of coordinated IEEE 802.11 localities, to evaluate the usability and effectiveness of the proposed middleware. The case study exemplifies how COSMOS supports articulated context-based access controls and shows how the different context evaluation strategies implemented allow the dynamic tuning of the middleware overhead depending on application- specific requirements.

2 COSMOS Security Framework

COSMOS is an access control middleware for securing agent-to-agent and agent-to-environment interactions in service provisioning scenarios with context awareness requirements. In particular, COSMOS focuses on three main peculiar aspects: flexible solutions for context-centric access control, active context view provisioning to middleware components, and privacy support in the propagation of user context information. COSMOS access control decisions depend on dynamic context attributes, such as resource state and availability, in addition to more traditional attributes, e.g., the identity of the MA code implementer, or the identity/role of the principal on behalf of whom MAs are executing.

Another distinctive feature of COSMOS is to provide MAs entering a new locality with a controlled visibility of the directly accessible physical/logical resources and of the other MAs locally executing (active context views). Active context views contain resources that both MAs are willing to access and the COSMOS access control function have qualified as accessible. The provision of active context views to MAs has many benefits. MAs can exploit the visibility of available resources to adjust their behavior accordingly and to reduce the risk of undesired task failure during execution. Active context views can also help MAs to decide whether it is more profitable to stay in a locality than to move from it and to explore a new computing environment.

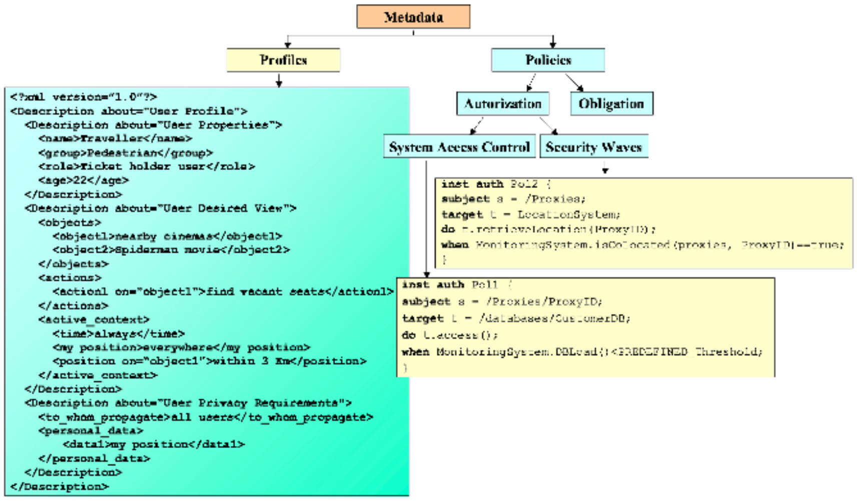
COSMOS addresses also the privacy issues that arise in context-aware service scenarios [13, 14]. In fact, context awareness requires computing devices that gather, collect and propagate up to the service level both user- and environment-specific information to take more informative service management strategies. However, the visibility of user-specific information, such as user location, could be exploited to infer user tasks or preferences, thus violating user privacy. COSMOS protects user privacy by enabling users to specify which personal context information they are willing to make public. User specifications guide and automate COSMOS access controls to user personal context information.

2.1 Security Model

To support context-centric access control, COSMOS allows system administrators and final users (on behalf of whom MAs act over the network) to specify their security requirements at a high level of abstraction in terms of metadata. Metadata are declarative rules that describe the attributes of users, devices and service components and the desired access control requirements.

A primary advantage of exploiting metadata is the possibility to separate security logic from security control. Metadata govern access control decisions, but are decoupled from the implementation of the system components in charge of enforcing access control accordingly to the metadata specifications. This favors the rapid prototyping of secure MA-based services, their runtime configuration and maintenance. By changing metadata to accommodate evolving security requirements, the behavior of a running agent system can be dynamically and rapidly varied without any intervention on agent and system code. In addition, developers do not need to manually insert calls to security checking code inside each resource that a host may want to protect from illegitimate agent usage. Metadata can also facilitate automated security reasoning: all basic elements involved in access control decisions can be easily extracted from declarative notations, analyzed and checked for conflicts. The relevance of metadata adoption to decouple management logic from mechanism implementation details has been recently recognized in network, systems and service management [15].

Figure 1 shows the different kinds of metadata supported in COSMOS, i.e., pro-files and policies. *Profiles* provide the explicit description of user/device/resource characteristics; in the following, we focus only on user profiles, while details on how device and resource profiles affect the context determination can be found in [4]. In particular, COSMOS decomposes the user profile in three sub-structures: user proper-ties, desired view, and privacy requirements. User properties specify user characteristics relevant to qualifying the user to the system, such as subscribed services, user identity or role.



```
<?xml version="1.0"?>
<Description about="User Profile">
  <Description about="User Properties">
    <name>Traveller</name>
    <group>Pedestrian</group>
    <role>Ticket holder user</role>
    <age>22</age>
  </Description>
  <Description about="User Desired View">
    <objects>
      <object1>nearby cinemas</object1>
      <object2>Spiderman movie</object2>
    </objects>
    <actions>
      <action1 on="object1">find vacant seats</action1>
    </actions>
    <active_context>
      <time>always</time>
      <my position>everywhere</my position>
      <position on="object1">within 3 Km</position>
    </active_context>
  </Description>
  <Description about="User Privacy Requirements">
    <to_whom_propagate>all users</to_whom_propagate>
    <personal_data>
        <data1>my position</data1>
    </personal_data>
  </Description>
</Description>
```

```
inst auth Pol2 {
subject s = /Proxies;
target t = LocationSystem;
do t.retrieveLocation(ProxyID);
when MonitoringSystem.isColocated(proxies, ProxyID)==true;
}
```

```
inst auth Pol1 {
subject s = /Proxies/ProxyID;
target t = /databases/CustomerDB;
do t.access();
when MonitoringSystem.DBLoad()<PREDEFINED_Threshold;
}
```

Fig. 1. Metadata taxonomy and examples in COSMOS.

Desired views express user preferences about the desired context information visibility (such as resources, other co-located users, . . .). For instance suppose that a user is willing to know if there are cinemas within 3 Km. As Figure 1 shows, his desired view is composed by the set of desired objects, i.e., cinemas (the `objects` tag), the set of actions that the user would like to perform on these objects, i.e., find vacant seats (the `actions` tag), and the context conditions in which the user desires to perform the specified actions, i.e., within 3 Km (the `active_context` tag).

Privacy requirements specify which personal information users are willing to propagate. This ensures a controlled disclosure of user-specific context information. In the profile example of Figure 1, the user allows the underlying system to propagate the information about her current position to all potentially interested users.

COSMOS adopts XML-based standard formats for profile representation to deal with heterogeneity of data representation over different architectures: standard XML for user profiles, the World Wide Web Consortium Composite Capability/Preference Profiles (CC/PP) for device profiles and the Resource Description Framework (RDF) for the interoperable description of resource components.

Policies are high-level directives regulating resource access control and defining choices in security management operations. COSMOS distinguishes between *authorization* and *obligation policies*. Authorization policies rule access control by defining what a subject can or cannot do on specific target resources if certain context conditions are met. These conditions may depend on the runtime resource/system state and on the subject identity/role. Obligation policies allow to automate security management tasks by specifying when the system must perform a specific management action on a set of target objects. For instance, MA system administrators can exploit obligation policies to block the execution of an agent when its CPU consumption is higher than a tolerated threshold. In the following, we focus only on authorization policies, essential for COS-

MOS context-centric access control model, whereas for details about obligation policies please refer to [16].

COSMOS distinguishes between system- and user-level authorization policies (*system access control* and *security waves* in Figure 1). The main difference is that the former ones are specified to protect system resources from illegitimate use by incoming MAs, the latter ones to reduce user privacy violations.

COSMOS exploits the Ponder language for expressing both types of authorization policies [17]. In Figure 1 `Pol1` is an example of system access control policy that allows an entering MA (the `subject` clause) to access the Customer database (the `target` clause) depending on the current database overload status (the `when` clause). Note that a subject can operate on target objects, by only invoking methods visible on the target interface and that the when clause allows to limit the applicability of authorization policies on the basis of context conditions. `Pol2` is an example of security wave allowing the propagation of the user location information to all interested users. Note that `Pol2` is the Ponder-based representation of the privacy requirements contained in the user profile shown in Figure 1. It is COSMOS that automatically generates `Pol2`: user privacy requirements are transformed into authorization policy rules that define which entities can have access to the user context information and under which circumstances. This transparent mapping relieves users from dealing with the details of the underlying system and facilitates the enforcement of user privacy requirements.

COSMOS exploits all the described metadata information to perform access control decisions and to determine the active context view to return to incoming MAs. In particular, as it stems from Figure 2, an active context view results from the intersection between a desired view and an allowed view. An allowed view contains the collection of local resources accessible to one MA depending on both system access control policies and security waves.

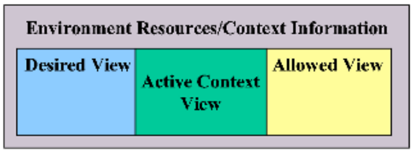


Fig. 2. Active context view.

3 COSMOS Middleware

COSMOS has been built on top of the Java-based SOMA system that supports the accessibility of mobile users/terminals to both traditional Web and to new context-dependent services. SOMA is centered on the distributed deployment of active middleware proxies over the fixed network to support service provisioning to portable devices. SOMA provides any portable device with a companion middleware proxy (*shadow* proxy) that autonomously acts on its behalf, possibly negotiates service tailoring to fit user/device characteristics and follows user/device movements among network localities. SOMA implements shadow proxies by exploiting the MA programming paradigm. In particular, SOMA provides proxies with execution environments, called places, that typically model nodes. Places can be grouped into domains that correspond to network localities, e.g., either Ethernet-based LANs or IEEE 802.11b-based wireless LANs. With a finer degree of detail, a shadow proxy is implemented by one SOMA agent running on a place in the domain where the portable device is currently located. SOMA domains can facilitate policy evaluation and enforcement for context-centric access control. In fact, the domain abstraction allows to define a well-specified management boundary: each domain holds references to the entities currently members of the domain (both MAs and resources) and to the applicable metadata. In particular, profiles and security waves are maintained in the SOMA directory service with global visibility and are accessible via the local domain directory component [6]; system access control policies are stored locally at the do-mains where they have to be enforced to increase management decentralization and local policy access.

At first proxy instantiation, COSMOS creates a secure trust relationship between the device and the proxy according to the protocol steps described in [18]. At runtime, proxy interactions with the hosting environments are protected by means of the COSMOS middleware facilities shown in Figure 3. The **Context-aware Security Manager** returns to proxies active context views on the basis of specified metadata, the **Authorization Enforcement Manager** performs context-centric access control decisions and the **Metadata Manager** enables users to specify the needed metadata. COSMOS facilities exploit SOMA lower-level functions for proxy identification, resource discovery, directory, context monitoring, and event registration/dispatching [6].

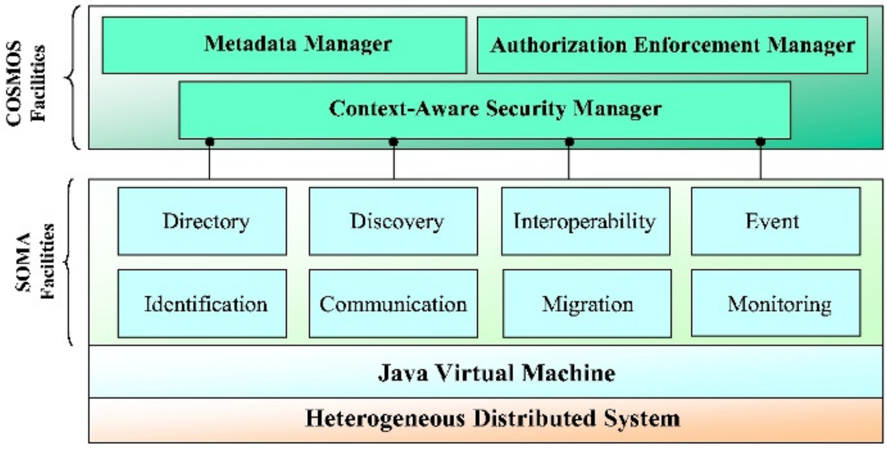


Fig. 3. COSMOS middleware facilities.

Metadata Manager (MM). MM provides various tools for metadata editing, updating, removing, and browsing. In particular, with regards to authorization policies, MM integrates tools for syntactic analysis of policy specifications, for translating user privacy requirements into security waves, and for transforming both system access control policies and security waves into Java objects, which act as policy information containers to be interpreted at runtime [16]. MM is also in charge of distributing specified metadata to the SOMA directory component responsible for storage, i.e., the one in the domain of first user registration (user home domain). At its first instantiation, any shadow proxy exploits the SOMA directory to retrieve its profile and security wave, which become part of its carried state.

Context-Aware Security Manager (CASM). CASM is responsible for establishing the active context view of any entering MA. When a new shadow proxy enters a domain, CASM calculates and returns to the proxy a valid active context view on the basis of the proxy responsible user profile, of the active system access control policies and of the security waves imposed by the other proxies currently executing in the domain. The active context view sent to the proxy is a copy of the active view maintained by CASM for any proxy in the locality, until the proxy exits from the domain.

CASM is also in charge of maintaining active context views up-to-date when relevant variations in context information occur, such as changes in the MAs executing in a locality, in resource availability, or in security wave specifications. CASM allows service providers to choose among differentiated view update strategies. At the two extremes, CASM supports eager and lazy update strategies. The *eager* strategy requires CASM to constantly coordinate with the underlying monitoring and event services in order to update active context views at the occurrence of any relevant context change. This strategy is expensive especially in environments with frequent context modifications, but provides proxies with an immediately and always updated active view. This is necessary when the updated active view is crucial to continuously re-adapt service tailoring/configuration strategies and to frequently evaluate the convenience to stay in the current domain.

The *lazy* strategy consists in updating the active context view only on user-demand, i.e., when the user requests the access to a resource in her current active view. This reduces CASM management load, but limits the possibility for a continuous service adaptation.

The choice of the most appropriate strategy to adopt depends on several factors, ranging from service requirements to the characteristics of the service deployment setting and to the tolerated trade-off between tailoring/configuration optimal choices and context update overhead. Similar considerations guided the implementation of protection domains in the JDK 1.2 security architecture [11].

Authorization Enforcement Manager (AEM). AEM mediates proxy-resource interactions by granting/denying proxies the access to resources, possibly depending on runtime conditions. Shadow proxies cannot directly access the resources included in their active context view, but have to interface with AEM at any resource access request. Active context views provide proxies with only the identifiers of accessible resources along with the permitted actions; no direct handles to the resources listed in the active view are returned to proxies.

When a proxy requests to access a resource, AEM intercepts the request and evaluates whether to deny/accept it. The type of access control checks needed depends on the update strategy adopted by CASM. In the case CASM adopts an eager update strategy, AEM grants/denies the proxy request by simply checking whether the requested resource and action are included in the already updated active context view maintained by CASM. Let us note that if CASM, at the moment of the access control check request, is engaged in a view update task, AEM waits until CASM terminates the update and then performs the requested check. The eager strategy can speed up access control checks: at the resource access request, AEM does not need to evaluate all the applicable specifications of system access control policies and security waves; CASM have already calculated all granted permissions to maintain a fresh active context view.

On the contrary, in the case of lazy update strategy, AEM has to calculate the set of permissions that currently applies to the requesting proxy. AEM has to evaluate all authorization policies specifications and also to coordinate with the monitoring system to check if the context information specified in the when clause holds in the system.

4 Case Study

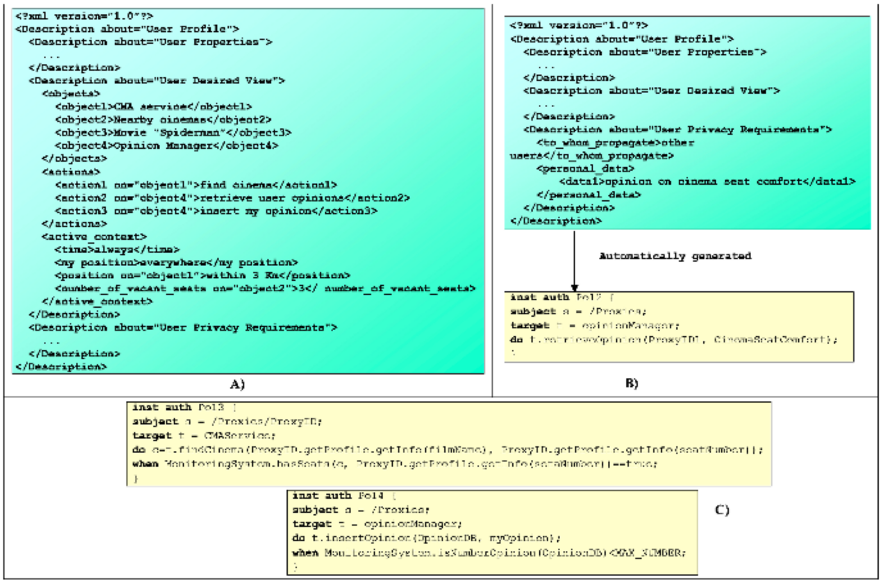
We have tested COSMOS in the design and implementation of a Context-centric Movie Assistant (CMA) that allows mobile users to find nearby cinemas and to exchange opinions about cinema characteristics, such as seat comfort, air conditioning, sound and screen resolution. The service exploits the visibility of user location to retrieve cinemas in the neighborhood performing the desired movie and the awareness of propagated user opinions to enable the choice of the most comfortable cinema according to the user-specific requirements.

Our testbed setting for CMA consists of a wireless metropolitan network composed by several 802.11 network localities, with each locality modeled as a SOMA domain. Each domain provides execution environments for shadow proxies, offers COSMOS middleware facilities and hosts info service components providing information about the locally available cinemas.

CMA users interact with the COSMOS infrastructure via device-specific clients running on their wireless access devices (Toshiba e740 Pocket PCs with Wi-Fi connectivity). The clients allow users to subscribe to CMA, by filling in a form that specifies user profile information and to authenticate to the service before starting any CMA session. After successful authentication, CMA instantiates a shadow proxy in the domain where the user is currently attached, and the proxy loads the user profile in its state part. At service provision time, the clients are only in charge of forwarding user requests (and of visualizing the received service results) to (from) their responsible proxies.

Let us consider the case of a user willing to see the “Spiderman” movie. As Figure 4A) shows, the user specifies in her user profile that she desires to know if there are nearby cinemas performing “Spiderman” and if they still have seats available. The user is also interested in using a local service of opinion management not only to browse other user opinions about the cinema characteristics but also to insert her personal judgment on a visited cinema.

Another user may have already visited one of the nearby cinemas and expressed an opinion about its characteristics. The excerpt of the user profile in Figure 4B) shows the



```
<?xml version="1.0"?>
<Description about="User Profile">
  <Description about="User Properties">
    ...
  </Description>
  <Description about="User Desired View">
    <objects>
      <object1>CMA service</object1>
      <object2>Nearby cinemas</object2>
      <object3>Movie "Spiderman"</object3>
      <object4>Opinion Manager</object4>
    </objects>
    <actions>
      <action1 on="object1">find cinema</action1>
      <action2 on="object4">retrieve user opinions</action2>
      <action3 on="object4">insert my opinion</action3>
    </actions>
    <active_context>
      <time>always</time>
      <my position>everywhere</my position>
      <position on="object1">within 3 Km</position>
      <number_of_vacant_seats on="object2">3</ number_of_vacant_seats>
    </active_context>
  </Description>
  <Description about="User Privacy Requirements">
    ...
  </Description>
</Description>
```

A)

```
<?xml version="1.0"?>
<Description about="User Profile">
  <Description about="User Properties">
    ...
  </Description>
  <Description about="User Desired View">
    ...
  </Description>
  <Description about="User Privacy Requirements">
    <to_whom_propagate>other
users</to_whom_propagate>
    <personal_data>
        <data1>opinion on cinema seat comfort</data1>
    </personal_data>
  </Description>
</Description>
```

```
inst auth Pol2 {
subject s = /Proxies;
target t = opinionManager;
do t.retrieveOpinion(ProxyID1, CinemaSeatComfort);
}
```

B)

```
inst auth Pol3 {
subject s = /Proxies/ProxyID;
target t = CMAService;
do c=t.findCinema(ProxyID.getProfile.getInfo(filmName), ProxyID.getProfile.getInfo(seatNumber));
when MonitoringSystem.hasSeats(c, ProxyID.getProfile.getInfo(seatNumber))==true;
}
```

```
inst auth Pol4 {
subject s = /Proxies;
target t = opinionManager;
do t.insertOpinion(OpinionDB, myOpinion);
when MonitoringSystem.isNumberOpinion(OpinionDB)<MAX_NUMBER;
}
```

C)

Fig. 4. User profiles and authorization policies.

privacy requirements of this user, which allow the system to propagate the opinion about cinema characteristics to all other users co-located in the domain.

Figures 4B and 4C) describe some authorization policies that rule the proxy visibility of local resources. Pol3 allows the proxy active context view to include nearby cinemas performing the desired movie if they have enough seats available. `Pol2` is the security wave that allows user opinion propagation. In particular, `Pol2` is automatically generated by COSMOS and derived from the user privacy requirements shown in Figure 4B). `Pol4` regulates proxy write operations on the opinion database managed by the opinion manager service. In particular, the policy states that all users can add their opinion to the opinion database if the number of already stored opinions does not exceed a threshold.

Given these configuration settings, when the proxy acting on behalf of the user represented by the profile in Figure 4A) enters the network locality, CASM retrieves and interprets the user profile and the policies to apply to the proxy, i.e., the policies where the proxy compares as the subject, and evaluates the policies by coordinating with the SOMA monitoring facility. On the basis of profile/policy metadata and of context conditions currently holding in the system, CASM generates the user active context view, composed by the nearby accessible cinemas with needed seat availability, by the local opinion manager service component where it is possible to retrieve the opinions about cinemas without violating user privacy, and by the list of actions that the proxy can perform on resources.

CASM can adopt various strategies to update the active context view. In the case of the eager strategy, CASM constantly monitors cinema seat availability and the status of the opinion database to check whether new opinions have been inserted. For instance, as long as the number of vacant seats is higher than the number requested by the user and no new opinions are available, the active context view does not change. Otherwise, CASM immediately updates the active context view, e.g., by removing from the nearby list of cinemas the one that have no more seats available. In the case of the lazy strategy, CASM calculates the active context view only once at first proxy arrival in the locality. Once the proxy receives the active view, it can decide the cinema to reach.

Let us note that the two update strategies have a different impact on the level of quality of service offered to the user. If CASM adopts the eager strategy, the user has online visibility of vacant seats in the cinema. This means that the user can see if other users are buying the last tickets and consequently can change her destination. On the contrary, if CASM adopts the lazy strategy, its management overhead is reduced, but the user could have an obsolete view of the cinema seat availability status. So she can run the risk of going to the cinema, of parking her car and of finding out only at destination that the needed tickets are no more available. Suppose now that at the end of the movie the user is willing to express her opinion about the cinema characteristics. Her device-specific client forwards the request along with the opinion content to the responsible proxy that tries to access the opinion data-base. AEM intercepts the request and Pol4 rules the proxy access to the opinion database. AEM access control checks depend on the active context view update strategy adopted by CASM. In the case of the eager strategy, CASM has already evaluated `Pol4` at proxy request time by maintaining the "insertOpinion" action in the list of permitted actions as long as the opinions in the database remain under the predefined maximum number. This relieves AEM from repeating policy evaluation: if the "insertOpinion" action is included in the fresh active context view maintained by CASM, AEM allows the proxy to insert user opinion. In the case of the lazy strategy, AEM has to evaluate Pol4 and check the conditions in the `Pol4` when clause.

5 Conclusions and Ongoing Work

Effective service provisioning over the wireless Internet requires the full visibility and the flexible management of context information. Requirements for context visibility start to be recognized in the security area also for traditional fixed networks, where interesting novel proposals are emerging to enhance protection techniques with context awareness [19, 20]. By focusing on mobile environments, COSMOS pro-poses and implements a novel security model for context-centric access control that exploits different types of metadata to express articulated security strategies, separately from the application logic. This separation of concerns increases flexibility, dynamicity and reusability of middleware/service components. In addition, the choice of the most proper evaluation strategy allows administrators to tune the access control overhead, which is acceptable for most wireless Internet services, usually having no hard real-time constraints.

First experiences in implementing services on top of COSMOS are encouraging further research to extend the middleware prototype. We are working on how to deal with possible runtime conflicts among policies to be enforced, and we are investigating and prototyping solutions based on policy prioritization. Finally, we are integrating COSMOS with mechanisms for inter-cell mobility prediction based on IEEE 802.11 signal strength variations, in order to anticipate the migration of (copies of) shadow proxies towards the new locality and the determination of the new active views in advance.

References

1. L.F. Akyildiz, J. McNair, J. Ho, H. Uzunalioglu, W. Wenye, “Mobility management in current and future communications networks”, *IEEE Network*, Vol. 12, No. 4, July/August 1998.
2. A.K. Dey, G.D. Abowd, “Towards a Better Understanding of Context and Context-Awareness”, *Proc. of CHI*, The Hague, The Netherlands, April 2000.
3. T. Rodden, K. Cheverst, K. Davies, A. Dix, “Exploiting Context in HCI Design for Mobile Systems”, *Proc. of Workshop on Human Computer Interaction with Mobile Devices*, Scotland, May 1998.
4. P. Bellavista, A. Corradi, R. Montanari, C. Stefanelli, “Dynamic Binding in Mobile Applications: a Middleware Approach”, *IEEE Internet Computing*, Special Issue on “Mobile Applications", Vol. 7, No. 2, March/April 2003.
5. IKV++ Technologies AG, enago Open Service Platform, http://www.ikv.de
6. P. Bellavista, A. Corradi, C. Stefanelli, “The Ubiquitous Provisioning of Internet Services to Portable Devices", *IEEE Pervasive Computing*, Vol. 1, No. 3, July-September 2002.
7. G. Vigna, (ed.), “Mobile Agents and Security”, Springer-Verlag, LNCS 1419, 1998.
8. R. Oppliger, “Security issues related to mobile code and agent-based systems”, *Computer Communications*, Elsevier Science, Vol. 22, No.12, 1999.
9. R. Montanari, C. Stefanelli, N. Dulay, “Flexible Security Policies for Mobile Agents Systems”, *Microprocessors and Microsystems*, Elsevier Science, Amsterdam, Olanda, Vol. 25, No. 2, 2001.
10. N. Mitrovic, U. Arronategui Arribalzaga, “Mobile Agent security using Proxy-agents and Trusted domains”, *Proc. of SEMAS 2002*, Bologna, Italy, July 2002.
11. L. Gong, “Inside Java 2 Platform Security”, Addison Wesley, 1999.
12. P. Bellavista, A. Corradi, C. Stefanelli, “Protection and Interoperability for Mobile Agents: A Secure and Open Programming Environment”, *IEICE Transactions on Communications (Special Issue on “Autonomous Decentralized, Systems")*, Vol. E83-B, No. 5, May 2000.
13. J. Al-Muhtadi, R. Campbell, A. Kapadia, M. D. Mickunas, S.g Yi, “Routing Through the Mist: Privacy Preserving Communication in Ubiquitous Computing Environments”, *Proc. of ICDCS’02*, IEEE Press, Vienna, Austria, July 2002
14. S. Wright, R. Chadha, G. Lapiotis (eds.), Special Issue on “Policy Based Networking”, *IEEE Network*, Vol. 16, No. 2, March 2002.
15. G. Myles, A. Friday, N. Davies, “Preserving privacy in environments with location-based applications", *IEEE Pervasive Computing*, Vol. 2, No.1, January/March 2003.
16. A. Corradi, N. Dulay, R. Montanari, C. Stefanelli, “Policy-driven Management of Mobile Agent Systems”, *Proc. of Policy 2001*, Springer-Verlag, LNCS 1995, Bristol, January 2001.
17. N. Damianou, N. Dulay, E. Lupu, M. Sloman, “The Ponder Policy Specification Language”, *Proc. of Policy 2001*, Springer-Verlag, LNCS 1995, pp. 18–39, Bristol, January 2001.
18. M. Winslett, T. Yu, K. E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, L. Yu, “Negotiating Trust on the Web", *IEEE Internet Computing*, Vol. 6, No. 6, November/December 2002.
19. M. J. Covington, W. Long, S. Srinivasan, A.K. Dey, M. Ahamad, G. D. Abowd, “Securing Context-Aware Applications Using Environment Roles", *ACM, SACMAT’01*, Chantilly, Virginia, USA, May 2001.
20. G. K. Mostéfaoui, P. Brézillon, “A Generic Framework for Context-Based Distributed Authorization”, *CONTEXT’03*, LNAI 2680, pp. 204–217, 2003.

Agent Support for Context-Aware Services and Personal Mobility

K. El-Khatib and G. v. Bochmann

School of Information Technology & Engineering, University of Ottawa
161 Louis Pasteur St., Ottawa, Ont., K1N 6N5, Canada
{elkhatib, bochmann}@site.uottawa.ca

Abstract. With the advent of smaller and ever faster computing and communications, more computing devices will be embedded in our life artifact. The explicit management of these embedded computers is definitely a burden for the regular person, especially when more than one device is required to complete a single task. This paper describes an architecture that integrates the use of agent technology, wireless communications and context-awareness to support service and personal mobility. A software agent, running on a portable device, leverages the existing service discovery protocols to learn about all services available in the vicinity of the user. The software agent runs a QoS negotiation and selection algorithm that sets the values for the session configurations parameters based on the session requirements, the user preferences, and the constraints of the devices running the services. The proposed architecture supports also service hand-off between devices to take account of service volatility in case of context change.

1 Introduction

Our work here is motivated by the growing need to provide context-aware service and personal mobility for roaming persons. Personal mobility [1] is defined as the ability of a user to get access to telecommunication services from any terminal (e.g. workstations, notebooks, Personal Digital Assistants (PDA), cellular phones) at any time and from any place based on a unique identifier of the user, and the capability of the network to provide services in accordance with the user's service profile. Closely related to the subject of personal mobility is service or session mobility [11], which refers to the ability of the user to suspend a service on a device and to pick it up on another device at the same point where it was suspended. An example of service mobility is a call transferred from the mobile phone of the user to his office phone.

The notion of context and its implications has been recently a research topic for a number of groups [2, 3, 4] and is still attracting more interest. The work has been mainly focusing on incorporating the context variables into the design of context-aware systems that dynamically respond to the need of its user. Context-aware variables might cover the physical, human, social, or organization elements [5, 6] of the context, to best fit the needs of its users.

One of the major enabling factors for context-aware systems is embedded computing, which is best described by Mark Weiser as the world with “invisible” machines; a world such that “its highest ideal is to make a computer so imbedded, so fitting, so natural, that we use it without even thinking about it”[7].

Another enabling factor for context-aware systems is the advance in short-range radio frequency communication, which created the notion of personal level communication infrastructure, also known as Wireless Personal Area Networking (WPAN). Bluetooth [8] is an example of WPAN technology. Connected over a WPAN, multiple devices can locate and communicate with each other. Due to their co-locality, these devices may harmonize collaboratively and provide services that cannot be possibly delivered with only one device. For instance, a video display service may use an audio playing service in its vicinity to play an audio and video recording. Also, an audio play-out service may use the service of a PC connected to the Internet to download and play music from the Internet.

Given these new trends in computation and communication, service and personal mobility come as a challenge to context-aware systems. The architecture proposed in this paper is an improvement on our personal mobility architecture [15], which was developed for static environments to make it support context change. It leverages technologies in short-range wireless communication, such as Bluetooth, to construct the WPAN; it also leverages service discovery protocols, such as Jini, SDP, and Salutation, to discover services available just in the WPAN of the roaming user. Running a service discovery protocol over a WPAN restricts the services available for the use of the user to devices that are within the WPAN and hence close enough to the user. The architecture supports also optional service mobility, which, if possible, allows services to follow the user across different contexts. A major component of our architecture is a *Personal Agent* (PA) process running on a personal device carried by the user. The PA triggers the service discovery and selection when it receives or initiates communication session requests; it also manages service mobility later on during the session.

The rest of the paper is organized as follows: Section 2 sets the stage with a usage scenario. Section 3 starts by presenting a literature review of a number of architectures for personal mobility with highlights to their limitations to static environments. We then propose our architecture for supporting context-aware personal mobility and its main components. After that, we present the algorithm used for service and Quality of Service (QoS) parameter values selection and explain how our architecture supports service mobility. We finally conclude in Section 4 and present a number of open questions.

2 Usage Scenario

The following usage scenario shows how the presented architecture could be used during a communication session between Alice, a team manager on a business trip, and her team-members. Alice would like to talk privately with Bob, who is a team leader in her group, prior to the meeting. Using the multimedia workstation in the business office of the hotel, Alice sends a communication session request to Bob 10 minutes before the meeting time. Bob, sitting in the lounge area, receives the call on his PDA for a multimedia conversation with Alice. Since Bob has indicated in his profile that he is always willing to accept calls from his manager Alice, Bob's PDA tries to find a microphone, a speaker, a video display service and a camera to make for a full multimedia session. Assuming that such services exist in the lounge area, the PDA can discover and reserve them for the communication session with Alice. The PDA then replies to Alice request with the addresses of the play-out services, and the full multimedia session is started between Alice and Bob.

When the time for the meeting comes, Bob moves with his PDA into the conference room where all the team members are waiting. Bob's PDA detects that the services that Bob was using are not available anymore, and since Bob has already enabled the follow-me option for the session, his PDA tries to discover similar services to continue the session in the conference room. The PDA then detects and selects the big screen, the camera, the speaker as well as the microphone of the conference room; Alice's picture appears on the big screen and she is now ready to participate in the meeting.

3 Architecture for Personal and Service Mobility

3.1 Related Architectures

A number of architectures [9, 10, 11, 12, 13, 14, 15] were proposed to solve the personal mobility problem in the Internet. All these architectures share the same concept of a Directory Service (DS) that provides a user-customized mapping from a unique user identifier to a registered device that is best suitable for the user to use. The basic idea of these architectures is that the user keeps a profile in the DS including static information about all the devices he/she uses, and the logic or policy for when to use these devices (user preferences). During session initiation, the DS executes the logic in the user profile and handles the communication request according to the user's specified preferences.

While these architectures provide personal mobility for users with multiple telecommunication devices (telephone, PDA, laptop, cellular phone, pager), they have a number of limitations:

- The information in the directory service is static,
- There is no support for service or device discovery in case of context change,
- They lack support for service mobility, and finally
- They lack support for complex (combined) services.

A number of researchers have addressed the problem of service selection in ubiquitous computing environments. The work in [16] presented two approaches for selecting services based on the physical proximity and line of sight of the handheld device relative to the service. The architecture suffers from the drawback of using infrared communication for finding and selecting services. The authors in [17] used a centralized approach where a central gateway is responsible for delegating rendering tasks to the devices in the environment of the user. A PDA carried by the user detects all devices in the surrounding, and sends the list to the central gateway to make the selection. In another work [18], the authors investigated the use of a browser running on a PDA to enable ubiquitous access to local resources as well as resources on the World Wide Web. The browser, called the Ubicompbrowser, detects devices and resources in the environment of the user, and delegates the rendering of the requested resources to these devices in order to overcome the limitation of the PDA. A major drawback of the Ubicompbrowser is that it uses a device directory that "stores all information about devices within the environment" [18] and requires the user to know its current location to restrict the set of available services. Service mobility and QoS issues are not discussed in either of these works, while they are centerpiece to our architecture.

The Virtual Home Environment (VHE) is another application that has been introduced in the third generation wireless mobile networks like the Universal Mobile Telecommunications System (UMTS) [19, 20]. VHE gives the user the ability to roam between different telecommunication networks and still receive the same set of services, thus giving the "feeling" of being on the home network. A number of researchers [21, 22, 23] proposed a Multi Agent System (MAS) for the provision and support of the VHE. In addition to seem-less access to subscribed services, the VHE provides also personalization of service to customers through adaptive terminals; our architecture goes beyond the VHE concept, which focuses mainly on Intelligent Network (IN) services. Our architecture focuses on services that are available to a user in a certain context, without a prior subscription to these services.

3.2 Proposed Architecture

Our work presented here is inspired by the Ubicompbrowser project, and is intended to support context-aware personal mobility. Our architecture builds on our previous architecture for personal mobility [15] and includes additional functionalities to overcome its limitation to static environments. The modified architecture uses the short-range Bluetooth wireless communication to construct the user's WPAN, and to restrict the domain of services available to the user just to the ones running on the devices that are within this WPAN. Our architecture differs also from the architectures in [16, 17] in that service selection is done automatically on behalf of the user without requiring the user to point and select each service individually using infrared, since the user might not (and should not) be aware of the services and their locations. We also address the problem of service mobility in dynamic context, by using periodical search for services similar to the services currently used by the user, in order to provide smooth hand-off for these services. We consider two services to be similar if they both serve the same purpose, for instance a TV and a wall projector, or a PC speakers and a mini-stereo.

Our previous architecture for personal mobility [15] is based on the concept of a Home Directory. The Home Directory (HD) has two functions: (a) storage of the user's profile and (b) forwarding of incoming communication requests. As a storage facility, the HD is a personalized database of users profiles, with each profile holding the user's contact information, current access information, preferences for call handling, presence service, authentication, authorization, and accounting information. With this logic stored with the data in the profile, end users can control the access permission to their devices according to their own preferences. To forward communication requests, a Home Directory Agent (HDA) executes the logic in the user profile every time the user receives an incoming call through the HD. The user can also invoke the HDA in order to find (and establish) the best way to place an outgoing communication request.

For a user in a dynamic context, the set of available devices for the user may change continuously as the context changes. Updating manually the information about currently available devices is not a feasible solution since the set of available services running on these devices changes continuously as the user moves from one location to the other. Additionally, detecting the devices and sending update messages to the HDA is not a practical solution since the change may occurs rapidly and constantly, which results in many update messages. Moreover, an update message might incur a certain delay so that by the time the message is received by the HDA, the information included in the message might be outdated.

To overcome these limitations, we propose to run a modified version of the HDA on a hand-held device, such as a PDA, that is carried by the user. We call this modified version of the HDA the *Personal Agent* (PA), and it is responsible for detecting devices in the vicinity of the user as well as managing the user's communication sessions. We require that the hand-held device, on which the *Personal Agent* runs, to have access to the Internet (through a wireless modem or IEEE 802.11[24] connection) in order to retrieve the user profile and send/receive communication requests through the HDA. The PDA is also supposed to be able to construct a Wireless Personal Area Network (such as Bluetooth WPAN) in order to be able to detect and communicate with other wireless devices just around the user. For the rest of the paper, we will assume that the PA is running on a PDA that satisfies these communication requirements.

At any one time, either the HDA or the PA is providing personal mobility service to the user. When the PDA is switched ON, the PA starts up and contacts the HDA to retrieve the user profile. From that point on until the PDA is switched OFF, the PA is responsible for executing the logic in the user profile, and the HDA would act only as a proxy for incoming call requests between the caller and the PA, forwarding all incoming requests to the PA. To ensure that the HDA is aware of the status of the PA, replies to communication requests are sent back through the HDA. If the HDA does not see a reply to a forwarded call after a certain time-out period, the HDA would know that the PA is not running or the PDA is currently out of reach. The HDA would then switch into active mode, and handle the request according to the rules specified in its local copy of the user profile.

The proposed *Personal Agent* is composed of five major components: a *Service Discovery Agent*, a *User Activity Monitor Agent* (UAMA), a *QoS Selection and Negotiation Agent* (QSNA), a *Service Registry*, and a *Communication Agent*. Fig. 1 shows the architecture of the *Personal Agent* with its components, followed by a detailed description of each of these components.

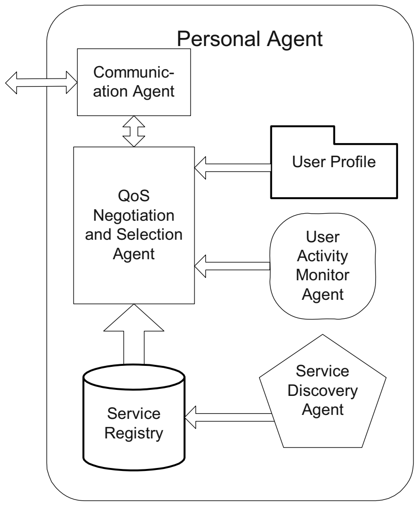


Fig. 1. Components of the *Personal Agent*

- *Service Discovery Agent* and *Service Registry*: The function of the *Service Discovery Agent* (SDA) is to search for all services in the WPAN of the user. The SDA provides the *QoS Selection and Negotiation Agent* (QSNA) (discussed below) with the list of currently available services. Since different services might be using different service discovery protocols (JINI [25], SDP [26], SLP [27], uPnP [28]), the SDA shall act as a service discovery client in multiple service discovery protocols. For service discovery protocols that support a central lookup service such as JINI, the SDA may query the lookup service for all possible services available for the user. For service discovery protocols that do not support the central directory service such as SLP without the directory agent, the SDA broadcasts in the WPAN a service discovery message to find the existing services. Once a session has been started, the SDA periodically searches for services that are similar to the services currently used by the session, and stores this information in the *Service Registry* (SR). The periodic update of this information in the SR allows for smooth service mobility for mobile users, as discussed in Section 2.4.
- *User Activity Monitor Agent* (UAMA): The UAMA keeps track of the current activity of the user (idle, busy…). The UAMA assists the QSNA during the service selection phase by providing up-to-the-minute information about the user's status. The UAMA can also incorporate additional information about the context of the user, such as information about whether he is by himself or surrounded by other people, as suggested in [18, 29].
- *QoS Selection and Negotiation Agent* (QSNA): Based on the session requirements, the information in the user profile, the list of available services provided by the SDA, and the user's current activity provided by the UAMA, the QSNA selects the best services that satisfy the session requirements, and complying to the preferences of the user (described in the user profile). The

QSNA might also mix and match several services to satisfy the requirements of the session. The QSNA implements the service and QoS parameter values selection algorithm presented in the next section.

– **Communication Agent:** The communication agent is responsible for receiving and sending communication requests from other communication agents. Our prototype uses the SIP [14] communication agent.

3.3 Algorithm for the Selection of Service and QoS Parameter Values

To provide personal mobility in a ubiquitous environment, a decision has to be made concerning (a) the service(s) (camera, speakers, microphone…) that should be used among the discovered services that meet the requirements of the communication session, and (b) what Quality of Service (QoS) parameter values (frame rate, frame resolution, audio quality…) should be used for each service. These decisions are affected by many factors including the media type of the exchanged content, the hardware and software profile of the devices where the services are running (also called device limitations), the status of the user, and finally the properties, preferences, and privileges of the end user (included in the user's profile). This process is carried out in two steps.

When the PA receives an incoming call through the HDA, and assuming that the user is willing to accept the call, the first step in the selection process consists of selecting the services that are required for establishing the communication session. Based on the session requirements, the QSNA queries the SDA for the necessary services for the session. For instance, a communication session that includes the receipt of audio media requires the discovery and selection of an audio playing service.

The second step in the selection process consists of selecting the QoS parameter values for each service, such as the audio quality for an audio service and the frame rate and resolution for a video service. This selection is based on the hardware/software limitations of the device, network condition, and the preferences of the user. For instance, the network access line (e.g. modem or wireless connection) of one of the devices may limit the maximum throughput of that device Also, a video content that requires a high-resolution display eliminates the possibility of using the display service of a monochrome device, especially if the user expressed his desire to receive only high quality video. This selection process also deals with the case when multiple similar services exist in the environment of the user. The QSNA always selects the services that result in a higher satisfaction value to the user. Additionally, the user may impose other limitations on the communication session by establishing a maximum cost per minute (which may depend on the effective network throughput used).

We base the selection of QoS parameters for each service on the concept of maximizing the user's satisfaction. In [15], we presented an extension to the work presented in [30], wherein, the QoS parameters for each service are selected based on a user satisfaction function. For each QoS parameter, the user indicates in his/her profile a satisfaction function that maps a QoS parameter value to a satisfaction value in the [0..1] range. The user also assigns a weight value for each QoS parameter. The total satisfaction value of the user for a combination of services is based on weighted combination of his/her satisfaction with each individual value parameter of the service. Using all possible combinations of QoS parameters of all available services, we select the combination that generates the maximum satisfaction within the restrictions of the device where the service is running and the preferences of the user. More details about the selection algorithm are given in [15].

3.4 Support for Service Mobility

During a communication session, a nomadic user might move away from one device hosting a service and get closer to another device that provides a similar service. Service mobility is required since the life span of the communication session might be longer than the time that the currently used device is available in the user's WPAN. When a device that is currently used becomes unavailable, the *Personal Agent* should switch to another device providing a similar service if available or it should inform the user about the disappearance of the service. For instance, a user moving away from his computer and entering the conference room should have, if desired, the multimedia session transferred from his computer to the TV and stereo system in the conference room. If the conference room does not have a TV set, the user should be warned that the video display service would be discontinued if he/she stays in the conference room.

Since our architecture is designed to support nomadic users, the architecture supports service mobility through service hand-off, transparently to the user. A smooth transparent service handoff requires continuous discovery of currently available services and updating of the service registry. When a communication session is in progress, the SDA periodically updates the information in the local service registry (SR). When a connection to a service is fading, the SDA informs the QSNA about a replacement service, and the QSNA switches quickly to the new service.

The *Personal Agent* may also use several heuristics during service selection such as selecting cheaper services, or selecting the services that have longer life span, even with lower quality, in order to avoid service disruption and to minimize the number of service hand-off.

4 Conclusion and Open Questions

In this paper, we have presented an architecture for supporting context-aware service and personal mobility. The architecture allows nomadic users to benefit from the availability of hidden services in a ubiquitous environment to establish communication sessions. To construct this architecture, we introduced a new component that we called the *Personal Agent* (PA) that acts on behalf of the user during the service discovery and selection process. The *Personal Agent* also provides support for service mobility through periodic updates of currently available services into a local service registry.

Presently, we have finished implementing the architecture and we are studying its feasibility and usability. Preliminary results show that session setup and service-switching time is between 5 and 55 *sec*, which are notably long. We noticed also that there is a different degree of tolerance between the audio and video switching: while most people do not mind a long service switching time for the video stream, they are fussy about the audio switching time.

During the design phase, we came across a number of open questions that are subjects for future research directions. For instance, the question of consolidating the service discovery transactions among many different service discovery protocols (such as Jini, SLP, SDP, Salutation,..) is still an important question that needs to be addressed in order to achieve usability and interoperability between these different protocols. Also, defining the scope of a service is essential in order to be able to determine when a user is out of the scope of a service and a replacement service should be used. The Cricket [31] and RAUM [32] projects have shown promising preliminary results on this subject.

Acknowledgment

This work was partially supported by CITO (Communications and Information Technology Ontario). The authors would like to thank Eric Zhen Zhang for his helpful comments and for implementing a prototype of the architecture.

References

1. H.Schulzrinne. Personal mobility for multimedia services in the Internet. European Workshop on Interactive Distributed Multimedia Systems and Services (IDMS), Mar. 1996.
2. Ubiquitous Computing (Xerox PARC). http://ubiq.com/hypertext/weiser/UbiHome.html
3. Contextual Computing Group (Georgia Tech). http://www.cc.hatech.edu/gvu/ccg/index.html.
4. Ambiante (Collaborative Buildings) (GMD). http://www.darmstadt.gmd.de/ambiente.
5. M.Beaudouin-Lafon and W.E.Mackay. Research Directions in Situated Computing. CHI'2000,The Hague (Netherlands), April 2000. ACM Conference on Human Factors in Computing Systems; 369, 2000.
6. B.MacIntyre. Context-aware personal augmented reality. CHI'2000,The Hague (Netherlands), April 2000. ACM Conference on Human Factors in Computing Systems; 369, 2000.
7. http://www.ubiq.com/weiser
8. J.Haartseen, M.Naghshineh, and J.Inouye. Bluetooth: Vision, Goals, and Architecture. ACM Mobile Computing and Communications Review, Vol. 2, No. 4, pp. 38-45, October 1998.
9. N.Anerousis et. al. TOPS: An architecture for telephony over packet networks. IEEE Journal of Selected Areas in Communications, Jan. 1999.
10. M.Roussopoulos, P.Maniatis, E.Swierk, K.Lai, G.Appenzeller and M.Baker. Person-level Routing in the Mobile People Architecture. Proceedings of the USENIX Symposium on Internet Technologies and Systems, October 1999.

11. Wang et al. ICEBERG: An Internet-core Network Architecture for Integrated Communications. IEEE Personal Communications Special Issue on IP-based Mobile Telecommunication Networks, 2000.
12. O.Kahane and S.Petrack. Call Management Agent System: Requirements, Function, Architecture and Protocol. IMTC Voice over IP Forum Submission VOIP97-010, 1997.
13. ITU-T Recommendation H.323. Packet-Based Multimedia Communications Systems. February 1998.
14. M.Handley, H.Schulzrinne, E.Scholler, and J Rosenberg. SIP: Session Initiation Protocol. RFC2543, IETF, March 1999.
15. X.He, K.El-Khatib, G.v.Bochmann. A communication services infrastructure including home directory agents. U. of Ottawa, Canada. May 2000.
16. M.Barbeau, V.Azondekon, R.Liscano. Service Selection in Mobile Computing Based on Proximity Confirmation Using Infrared. MICON 2002.
17. G.Schneider, C.Hoymann, and S.Goose. Ad-hoc Personal Ubiquitous Multimedia Services via UpnP. Proceedings of the IEEE International Conference on Multimedia and Exposition, Tokyo, Japan; Aug. 2001.
18. M.Beigl, A.Schmidt, M.Lauff, H.W.Gellersen. The UbicompBrowser. Proceedings of the 4th ERCIM Workshop on User Interfaces for All, October 1998, Stockholm, Sweden.
19. Mobilennium – The UMTS Forum Newsletter, No. 5 Nov (1998)
20. DTI, Developing Third Generation Mobile and Personal Communications into the 21st Century, 31 May 1997, http://www.gtnet.gov.uk/radiocom
21. S.Lloyd, R.Hadingham, and A.Pearmain.Virtual Home Environments Negotiated by Agents. 2nd International ACTS Workshop on Advanced Services in Fixed and Mobile Telecommunications Networks, CWC, Singapore, 1999.
22. http://www.uk.infowin.org/ACTS/RUS/PROJECTS/ac341.htm
23. T.Wierlemann, T.Kassing, B.Kreller. Usability of a Mobile Multi-Media Service Environment for UMTS. 3rd Acts Mobile Communication Summit, June 8-11,1998, V.1 p151-166.
24. IEEE802.11. http://grouper.ieee.org/groups/802/11/main.html
25. JINI (TM). Http: //java.sun.com/product/JINI/. 1998.
26. Bluetooth Special Interest Group (SIG). Service Discovery Protocol. SIG Specification version 1.0, Volume 1 Core, part E, pp 233-384.
27. E.Guttman, C.Perkins, J.Veizades, and M.Day. Service Location Protocol. Version 2. http://ietf.org/rfc/rfc2608.txt.
28. Universal Plug and Play Forum. Universal Plug and Play (UPnP). http://www.upnp.org
29. E.Horvitz, A.Jacobs, and D.Hovel. Attention-Sensitive Alerting. Proceedings of UAI ‘99, Stockholm, Sweden, July 1999, pp. 305-313. San Francisco: Morgan Kaufmann.
30. A.Richards, G.Rogers, V.Witana, and M.Antoniades. Mapping User Level QoS from a Single Parameter. Second IFIP/IEEE International Conference on Management of Multimedia Networks and Services, Versailles, November 1998.
31. N.B.Priyantha, A.Chakraborty, H.Balakrishnan. The Cricket Location-Support system. Proceedings of the 6th ACM MOBICOM, Boston, MA, August 2000.
32. F.Hupfeld and M.Beigl. Spatially aware local communication in the RAUM system. Proceedings of the IDMS, Enschede, Niederlande, October 17-20, 2000, pp 285-296, ISBN 3-540-41130-5

Delivery Context Negotiated by Mobile Agents Using CC/PP

Rosa Gil, Roberto García, and Jaime Delgado

Universitat Pompeu Fabra (UPF), Departament de Tecnologia,
Pg. Circumval·lació 8, E-08003 Barcelona, Spain
{rosa.gil,roberto.garcia,jaime.delgado}@upf.edu

Abstract. Our approach to content negotiation is a framework of mobile agents, where the agents can migrate from user devices to negotiation servers in order to get access to more resources. We took this approach and now we introduce new features in the architecture. The key idea is content customisation depending on device description with CC/PP (Composite Capability/Preference Profile). The objective is twofold. First, to improve consumer experience adjusting contents to consumption devices. Second, to rationalise network and device use spending only the necessary resources.
Altogether, it is a new step in the direction marked by the use of mobile agents in mobile devices. This way, computation and bandwidth consumption can be moved out of mobile devices to network devices, where these resources are cheaper.
Moreover, in contrast to direct browser-server content negotiation, our agent based negotiation framework provides independence between content negotiation and its consumption, i.e. content can be negotiated and experienced in different user devices, thus better adjusting to user preferences.
All this would be especially relevant when third generation (3G) mobile devices are widely available and more sophisticated multimedia content is available in mobile delivery contexts.

1 Introduction

First of all, we introduce our Mobile Agents Negotiation Framework. The work presented in this paper is build upon this framework in order to manage heterogeneous digital content “delivery contexts”. The next introduction subsections cover the framework and the extensions.

Then, in Sections 2 and 3, we go into detail about how to describe these “delivery contexts” and how to make the negotiation process “context aware”. Finally, in Section 4, the conclusions about current work and future plans on the integration of this framework with other technologies are presented.

1.1 Mobile Agents Negotiation Framework

As we developed in previous paper [1], mobile agents are a key aspect in order to achieve a flexible and robust generic negotiation architecture, where users are represented by agents who can migrate to negotiation servers as we see in Fig. 1. Once there, they can negotiate improving messages quantity interchange and computational power.

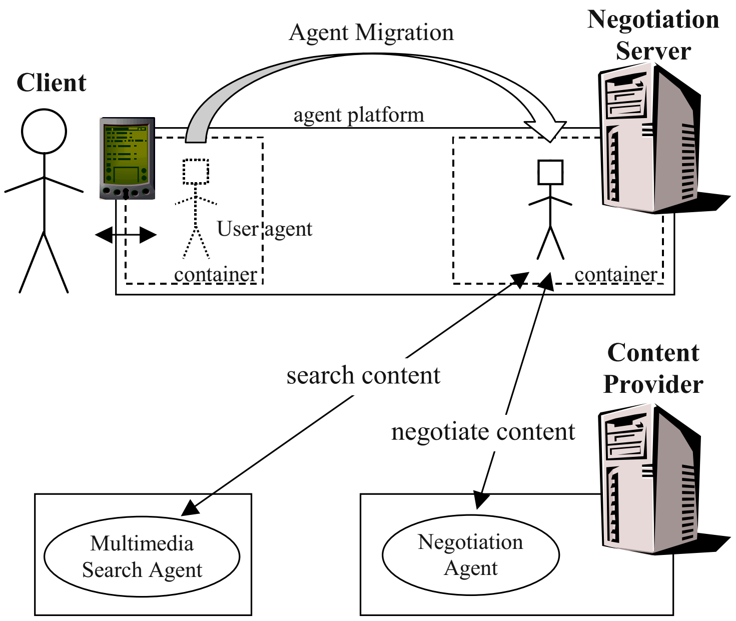


Fig. 1. Negotiation mobile-agent scenario

These ideas have been developed using JADE-LEAP [2] agents and intra-platform mobility. The framework comprises a set of containers and their type adjusts to the hardware platform that gives them support.

There is a main container hosted by the Negotiation Server suited for Java2 SE [3]. For mobile devices, the choices are container for PersonalJava [4] and light containers for Java2 ME and MIDP [5].

Mobility is managed from a rules inference engine implemented with Jess located at the main container. It contains a set of rules that move agents when it is convenient and balance the load of the different containers.

For instance, when a mobile agent wants to start a negotiation, it is moved to the main container at the Negotiation Server. There, the possibly intense negotiation message exchange and reasoning process support can be developed using more appropriate computational and network resources. More details are given in [1].

At the bottom of Fig. 1, there are the multimedia specific parts. They are being implemented as part of the NewMARS project [6]. It is a digital content portal with agent negotiation support. Its semantic approach, based on using ontologies like an Intellectual Property Rights one called IPROnto [7], provides means to enable advanced agent based negotiation of digital contents.

1.2 Improving the Architecture

The architecture depicted in the previous section is only the first step towards a mobile and transparent content management platform.

In this section the new features that have been planned are introduced. They are basically necessary because delivery contexts are becoming more and more heterogeneous. However, users do not want to have to deal with different particularised interfaces.

Mobile agents can become this homogeneous content management system and provide the means to make content negotiation and customisation particularities totally transparent to final users.

Moreover, at the same time that delivery contexts diversity increases, new specialised technologies appear. They are related to delivery features but also to other accessory ones with which our mobile agents solution is required to interact. Therefore, we also present our preliminary explorations of an interoperability framework for mobile agents solutions.

More details about all these ideas are presented in the next subsections and summarised in the new scenario presented in Fig. 2.

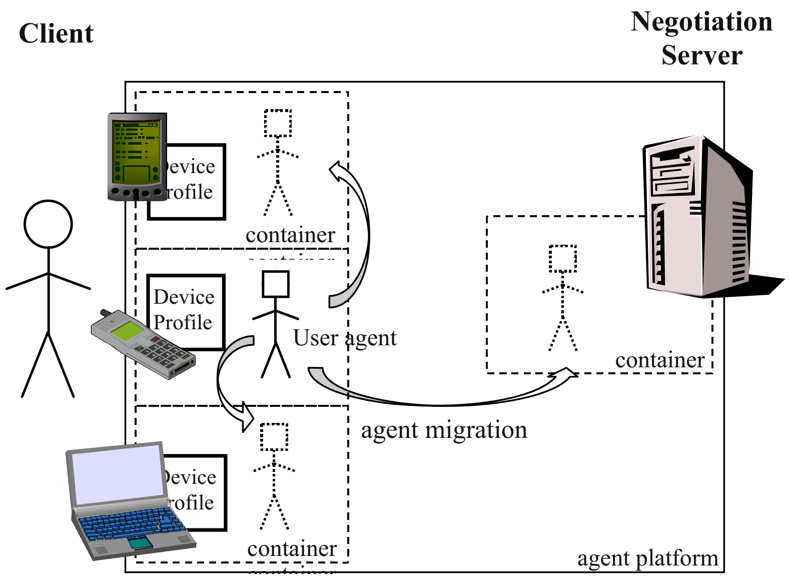


Fig. 2. Negotiation mobile-agent platform supporting multiple user devices

1.2.1 Multiple Devices

As we have already pointed out, there is an increasing range of final users devices that cannot be translated to an even more heterogeneous set of interfaces for them. Mobile agent can help users deal with this diversity but, in order to do that, mobile agents must be aware of the kind of devices they must deal with.

Therefore, mobile agents need device descriptions where their characteristics are explicitly stated. There is a W3C [8] initiative that is specifically conceived for this task, the CC/PP [9] one. A more detailed description of CC/PP is presented in Section 2.1.

The mobile agent knows the devices it is in charge of through the corresponding CC/PP profiles. They are explicitly stored in agent's knowledge base or implicitly, by reference. More details about how this is implement in our mobile agents appear in Section 2.3.

1.2.2 Content Customisation

Once the mobile agent is aware of the different delivery context it manages for its human user, it must take them into account when it is negotiating content.

The mobile agent starts negotiating with a content provider and informs it about the delivery context of the content. This can be done sending explicitly the content of the corresponding CC/PP profile or a reference of it, eventually augmented by the concrete device particularities.

Using the device profiles, the content provider can adjust its offer to the requested delivery context. On one hand, it can adjust context characteristics, for instance image size, colour depth, streaming bandwidth, etc. On the other hand, it can adjust economical and utilisation conditions accordingly.

Besides that, the existing negotiation protocol stays unaffected. The user mobile agent can directly accept the proposed conditions or start a negotiation cycle that approaches them to those it thinks its user may require.

A detailed view of how customisation is managed and how device profiles are used during negotiations is shown in Section 3.

1.2.3 Integrating with Future

We are looking at future tendencies. An important task will be to integrate mobile agents platforms with other initiatives. For instance, P2P, Web Services, Grid… The details are discussed in section 4.

2 Describing Delivery Contexts

A delivery context could be defined as a set of attributes that characterises the capabilities of the access mechanism and the preferences of the user. An access mechanism is a combination of hardware (including one or more devices and network connections) and software (including one or more user-agents) that allows a user to perceive and interact with the Web using one or more interaction modalities (sight, sound, keyboard, voice, etc.). Web servers and applications that adapt content for multiple access mechanisms must be sensitive to delivery context. Composite Capability/Preference Profile (CC/PP) corresponds to the W3C initiative in this line.

2.1 CC/PP Composite Capability/Preference Profile

CC/PP Composite Capabilities / Preferences Profile (CC/PP) provides a standard way for devices to transmit their capabilities and user preferences. It was originally designed to be used when a device requests web content via a browser so that servers and proxies can customize content to the target device, i.e. support device independence. It is a proposed industry standard for describing delivery context.

Moreover, the World Wide Web Consortium (W3C) hosts two activities that are relevant to this specification. The first of these is the Device Independence Working Group, which is part of the W3C Interaction domain. This group has produced a document describing device independence principles [10] and two informal drafts, one discussing delivery context [11], i.e. mechanisms like CC/PP, and the other one discussing authoring [12].

The second relevant activity hosted by the W3C is the CC/PP Working Group, which is also part of the W3C Interaction domain [9].

UAProf [13], User Agent Profile, is a concrete implementation of CC/PP developed by the Open Mobile Alliance (OMA) that like the W3C is organised into areas or groups and UAProf is part of the Mobile Application Group. UAProf is an implementation of CC/PP aimed at WAP-enabled [14] mobile terminals.

CC/PP profiles contain capability and preference information sent from a client to a server. In order to validate CC/PP profiles, there must be a set of rules that determine what constitutes a valid profile. According to the CC/PP structure and Vocabularies Working Draft a CC/PP profile must first meet the constraint of being a valid XML and Resource Description Framework (RDF) document [15]. The W3C's RDF validation service [16] can be used to validate a profile in this way.

Currently there are two protocols developed for CC/PP exchange based on HTTP, CC/PP-ex [17] and W-HTTP, the last one based on WAP [18]. These protocols have many common features. They send CC/PP information in two forms: references profiles and profile diffs. A reference profile is sent as a URI between the client and the server. The server then uses this URI to retrieve the profile from a third source known as a profile repository. Profile diffs on the other hand are sent as in-line XML fragments and may or may not be present in the headers. Profile diffs are associated with a sequence number that indicates processing order.

2.2 Processing CC/PP

A first step in processing CC/PP is to make the current generation presentation-oriented Web technology interoperable with the next-generation Semantic Web technology [19]. For example, CSS [20] style sheets are currently not able to take CC/PP profiles into account. CSS has, however, a feature that is closely related to CC/PP, and allows the specification of device dependent style rules. Fig. 3 shows an example of a style sheet that uses smaller fonts on mobile devices screens than desktops screens for the same document.

```
@media screen { min-width:32px
body { font-size: 8 pt }
}
@media screen { min-width:640px
body{font-size: 12pt}
}
```

Fig. 3. Device dependent style rules as CSS3 extension

Style engines need to be able to deal with these features in order to take full advantage of the information specified in CC/PP delivery contexts.

Note that the need to take CC/PP information into account also applies to XSLT [21] transformation engines. One could, for example, imagine an extension of XSLT's mode concept. For example, transformation rules could be selected in a way similar to that of the media rules in CSS. In such a hypothetical extension, see Fig. 4, one could, for instance, define a rule for creating a two column layout only if the output medium is a desktop and the screen is wider than 1024 pixels.

```
<xsl:templatematch="body"
mode="screenand(min-width:1024px)">
...
<fo:region-bodycolumn-count="2"/>
...
</xsl:template>
```

Fig. 4. Using XSLT transformation engines

In addition to taking information about delivery contexts into account, style sheets also need to take into account the semantic information that is contained in the metadata associated with the content. Currently, style selector mechanisms only match on the syntactic properties of the underlying XML document hierarchy. This applies both to the selector mechanism used by CSS and to the XPath [22] selectors used by XSLT.

In all examples above, the rules were intended to match on the <body> element of an HTML document. However, to use the current generation CSS and XSLT engines to process general metadata is not practical to match on the semantic properties of metadata. For CSS and XSLT processors, RDF is just XML. As a result, it is very hard to write, for example, a rule that matches on all alternative XML serializations that are allowed for RDF [15].

A more serious problem, however, is that it is impossible to write CSS or XSLT rules that make use of the semantic features of RDF Schema [23] (RDFS). For instance, a style rule that applies to all objects that are instances of a specific RDFS class.

Future semantics-aware selector mechanisms would allow specification of rules in terms of the RDF semantics expressed in the metadata. This would extend the currently used CSS and XPath selectors, which are based on the XML syntax encoding the semantics. Consider the extended XSLT example rule in Fig. 5, which uses the RDF-aware query language RQL [24] for its selector, instead of XPath.

```
<xsl:template match="RQL(http://dmag.upf.es/schema.rdf#VideoMatrix)">
...
</xsl:template>
```

Fig. 5. Semantic matching of XSLT rules using RQL selectors

It matches on all resources that are instances of (subclasses of) the RDF class VideoMatrix. Given the fact that our RDF Schema would define "Matrix Reloaded" as a subclass of VideoMatrix, the rule would also match on the other HTML fragments. Such rules that employ the semantic relations defined in the metadata are currently impossible to write in XSLT.

2.3 Using CC/PP

Some API's as DELI (A Delivery context Library for CC/PP and UAProf) and Intel ® CC/PP SDK [25] have been implemented in the latest times to solve some of the previous problems and content negotiation has been implemented using CC/PP and WAP UAProf [26]. However, there is not a standard way of doing all this, but it is under development [27].

Meanwhile, we have preferred to use the available mobile agents communication facilities, i.e. FIPA-ACL messages. The mobile agent knows the devices it is in charge of through the corresponding CC/PP profiles. They are explicitly stored in agent's knowledge base or implicitly by references, URL, that point to the WWW location where they are stored. Moreover, there can be particular modifications to these profiles, for instance a device with an upgraded amount of memory.

In the next section, it is shown how agents interchange CC/PP profiles and use them during negotiation.

3 Negotiation with Customisation

Now, we go into the implementation part. First, in Section 3.1, the negotiation process implemented by mobile agents is presented. This process is extended, in Section 3.2, with CC/PP profile exchange and used during the negotiation process.

3.1 Negotiation Process

In the beginning, the mobile agent resides at one of the user devices, which is mobile agents capable, i.e. it has a mobile agents container installed. The user interacts through the agent's GUI to determine all the criteria required to select the content he is interested in. Then, the agent enters the search phase and it migrates to another agent-platform container. This new container is one of the server types, where it has better Internet connectivity and processing power. Thanks to these greater resources, it can carry on less constrained searches and a more accurate negotiation process of the required content.

When the user agent has enough information, it can start the automatic negotiation process through a protocol defined by rules. The retrieved licensing agent, who is in charge of negotiating the access to the desired content, is contacted and a call for proposals is issued. An initial offer is received, if the licensing agent really has the requested content. From this point, some counter-offers may be interchanged till the negotiation ends due to a reject or an agreement. During this negotiation the content is adjusted to the particular requirements of the delivery context, and consequently the corresponding commercialisation condition.

The negotiation results are then communicated to the user. To facilitate user interaction, the user agent returns to the agent-platform container at the user's mobile device.

3.2 Negotiating Customisation

The first change to allow negotiating customisation is to make the content provider agent aware of the delivery context conditions. ACL messages with the "inform" communicative act are used for this [28]. An example is shown in **Table 1**.

Table 1. Delivery context implicit exchange by an inform-ref preformative ACL message, example fragment extracted from the Siemens S55 mobile phone CC/PP profile available at http://communication-market.siemens.de/UAProf/S55_00.xml

```
(inform
   :sender User1MA
   :receiver ContentProviderA
   :content (<?xml version="1.0" encoding="UTF-8" ?>
      <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:prf="http://www.wapforum.org/profiles/UAPROF/ccppschema-20010330#"
           xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
       <rdf:Description ID="S55_Profile">
              <prf:component>
                 <rdf:Description ID="HardwarePlatform"> ...
              <prf:component>...
      </rdf:Description>
       </rdf:RDF>)
   :language fipa-rdf0 )
```

More specifically, the "inform" communicative act is used to communicate explicit delivery context information, i.e. FIPA-ACL message content is the CC/PP profile serialised in its RDF/XML form [29]. To communicate it implicitly, the "inform-ref" communicative act is used and the ACL message content is the URL reference pointing to the CC/PP online version. Although the content is in RDF/XML form, we have also inspired ourselves on the FIPA device ontology [30].

The use of RDF/XML profile encoding allows a direct integration of all this information in the negotiation process. DAMLJessKB [31] allows adding and extracting facts from Jess engine. In other words, RDF (including CC/PP delivery context profiles) is incorporated into the Jess engine.

Then, the negotiation policies are implemented by Jess rules, which can be exported to a common rules interchange format, RuleML[32], and also imported. The rules take into account delivery context information to steer the negotiation. For an example see **Table 2**.

Table 2. Jess rule that uses RDF metadata imported by DAMLJessKB to detect image support

```
(defrule are-images-supported
   (PropertyValue ?resource
      http://www.w3.org/1999/02/22-rdf-syntax-ns#type
      http://www.wapforum.org/.../UAPROF/ccppschema-20010330#HardwarePlatform)
   (PropertyValue ?resource
      http://www.wapforum.org/.../UAPROF/ccppschema-20010330#imageCapable "Yes")
=>
   (assert (include-images)))
```

4 Future Work

In this final section we present some future plans that try to solve the problems we have found when trying to integrate different agent technologies.

4.1 Mobile Agents Integration

During the development of the Mobile Agents Negotiation Framework, we have found some tasks that are not well faced by mobile agents solutions. For instance, negotiation decision tasks that are computation intensive. We have considered more convenient to encapsulate all these intensive computations as independent services and implemented by other technologies, for instance Grid networks [33].

At this point we have to face interoperability problems between this two different kinds of mobile computation networks. The first one, which we have used till now, seems more appropriate for user mobility environments as the one resolved by the previous content negotiation architecture. The second one, the Grid, appears as the best choice in the "server" side. When intensive computations are required the best choice is to transparently integrate many computational resource into a Grid. Inside this Grid, the required computations can be distributed to attain the best affordable computational throughput.

Therefore, as we have seen that combining both mobile code solutions is the best option, an interoperability layer is necessary. In Section 4.1.1, we present our preliminary integration architecture.

We propose to use a Peer-to-Peer solution in order to transparently and dynamically integrate both. Peer groups are configured dynamically from global UDDI repositories using Web Services tools. Once the required services have been found and configured, the ad-hoc peer group is established.

4.1.1 Peer-to-Peer Interoperability Layer

Our new proposed architecture model, presented in Fig. 6, shows how the different technological pieces are combined to meet the requirements. These pieces are Grid Computing, P2P, Semantic Web Services and Agents. These technologies are organized into a layer cake design.

At the bottom layer, there are the computational and storage resources managed by the Grid. This layer contains a set of grids that conform resource-sharing spaces with a unique logical access point. The problem is that users see these spaces like isolated islands, and, altogether, they behave like static groups of resources that must be put together by hand.

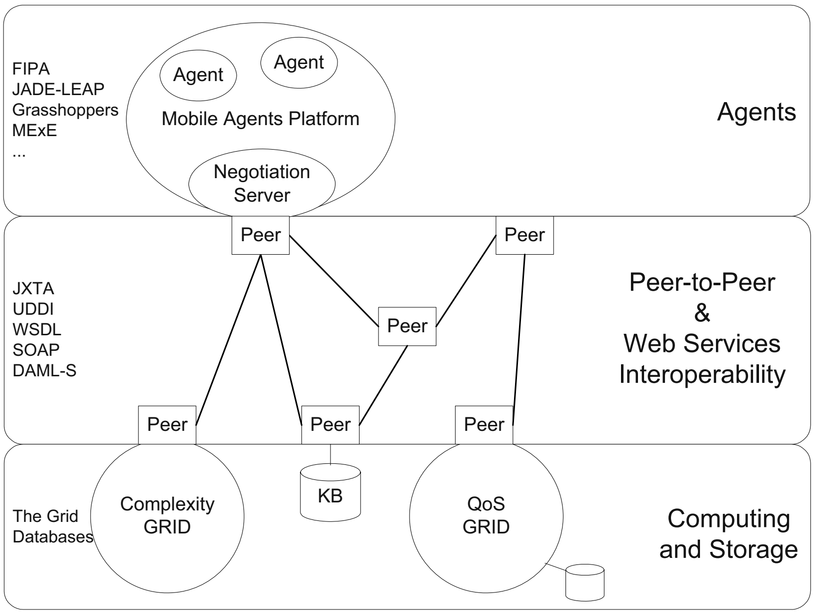


Fig. 6. Architecture model

Peer to peer technologies are the medium that enables an open communication among totally distributed resources. When P2P technologies are deployed over the Internet, a potentially global communication space is obtained. Therefore, P2P has been stacked over the Grid layer, more concretely the JXTA[34] package.

JXTA provides the building blocks to construct a network of peers over the Internet infrastructure using Web technologies. Each peer sets an access point to the resources it manages and, at the same time, is the medium to reach other peers. This feature has been used and a peer point has been associated to logical Grid access points.

5 Conclusions

A great variety of mobile devices are spreading over the world, but they are not the only ones, they share communications environment with desktops and laptops among other devices. An end user can have many of them; customisation is the only way to get desired content almost everywhere. So if we want to design a true service negotiating multimedia content, customisation is a goal to achieve. CC/PP seems to be a veritable tool and we have reviewed how it could be useful.

However, there is still a lot of work to do in order to integrate the semantic capabilities of CC/PP in the customisation process. We have outlined a server side specific option based on the use of the Jess rule engine with RDF semantics capabilities. However, in order to see a spread adoption of CC/PP, semantic capabilities should be integrated in style sheet technologies, i.e. CSS or XSLT.

Moreover, thinking about the nearest future it is important to have in mind a global idea about the network, because customisation implies to be capable of working in many technologies, as P2P or Grid presented in Section 4. Thus, we want to propose a complete business solution based on mobile agents. They can travel around Mobile Agent Platforms and negotiate in our name.

References

1. Delgado, J.; Gallego, I.; García, R. and Gil, R.: An Architecture for Negotiation with Mobile Agents. Mobile Agents for Telecommunication Applications, MATA 2002. Lecture Notes in Computer Science, Vol. 2521. Springer-Verlarg (2002) 21-31
2. JADE-LEAP, http://leap.crm-paris.com
3. Java 2 Platform Standard Edition, http://java.sun.com/j2se
4. Pjava, Personal Java. http://java.sun.com/products/personaljava/pj-emulation.html
5. Mobile Information Device Profile, http://java.sun.com/products/midp
6. AREA 2000 Spanish Government Research Project (TIC2000-0317-P4-05), NewMARS subproject, http://dmag.upf.es/newmars
7. Delgado, J., Gallego, I., Garcia, R., Gil, R.: An ontology for Intellectual Property Rights: IPROnto. Extended poster abstract, International Semantic Web Conference (2002) http://dmag.upf.es/papers/ExtAbstractPosterISWC.pdf
8. World Wide Web Consortium, http://www.w3c.org
9. Composite Capabilities/Preferences Profile Working Group, http://www.w3.org/Mobile/CCPP
10. Device independence principles, http://www.w3.org/TR/2001/WD-di-princ-20010918
11. Discussing delivery context, http://www.w3.org/TR/2002/WD-di-dco-20021213
12. Discussing authoring, http://www.w3.org/TR/2002/WD-acdi-20021018
13. DELI: A Delivery context Library for CC/PP and UAProf. http://www-uk.hpl.hp.com/people/marbut/DeliUserGuideWEB.htm
14. WAP forum. http://www.wapforum.org
15. Lassila, O., Swick, R.R. (editors): Resource Description Framework (RDF), Model and Syntax Specification. W3C Recommendation 22 February 1999 http://www.w3.org/TR/REC-rdf-syntax
16. RDF validation service. http://delicon.sourceforge.net/

17. Ohto ,H. and Hjelm, J.: CC/PP exchange protocol based on HTTP Extension Framework. W3C Note 24 June 1999, http://www.w3.org/TR/NOTE-CCPPexchange
18. WAP Forum Releases, Technical Specification, http://www.wapforum.org/what/technical.htm
19. Garcia, R.; Delgado, J.: Brokerage of Intellectual Property Rights in the Semantic Web. 1st Semantic Web Working Symposium (SWWS-1), Stanford, CA (2001)
20. Wugofski, T., Dominiak, D., Stark, P. and Roy, T.: CSS Mobile Profile 1.0. W3C Candidate Recommendation 25 July 2002, http://www.w3.org/TR/css-mobile
21. Clark, J. (ed.): XSL Transformations (XSLT) Version 1.0 W3C Recommendation 16 November 1999, http://www.w3.org/TR/xslt
22. XML Path Language, http://www.w3.org/TR/xpath
23. Brickley, D. and Guha, R.V. (eds): RDF Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft 30 April 2002. http://www.w3.org/TR/rdf-schema
24. Karvounarakis, G.; Alexaki, S.; Christophides, V.; Plexousakis, D.; Scholl, M.: RQL: A Declarative Query Language for RDF. The Eleventh International World Wide Web Conference, 2002
25. M.Bowman, R.D. Chandler, D V. Keskar. Intel Labs. Delivering Customized Content to Mobile Device Using CC/PP and the Intel ®CC/PP SDK.
26. Butler M. H.: Implementing Content Negotiation using CC/PP and WAP UAProf. Hewlett-Packard (2001)
27. Composite Capability/Preference Profile (CC/PP) Processing Specification. Public Draft, version 0.5 (2003)
28. FIPA Communicative Act Library Specification, http://www.fipa.org/specs/fipa00037/SC00037J.html
29. Adapted Content Delivery for Different Contexts http://opera.inrialpes.fr/people/Tayeb.Lemlouma/Papers/saint2003.pdf
30. FIPA Device Ontology Specification. http://www.fipa.org/specs/fipa00091/XC00091C.pdf
31. DAMLJessKB documentation. http://edge.mcs.drexel.edu/assemblies/software/damljesskb/javadocs/edu/drexel/gicl/daml/damljesskb/DAMLJessKB.html
32. RuleML http://www.semanticweb.org/SWWS/program/full/paper20.pdf
33. Foster, I.; Kesselman, C.; Tuecke, S.: The Anatomy of the Grid, Enabling Scalable Virtual Organizations. International J. Supercomputer Applications, 15:3 (2001)
34. JXTA project, http://www.jxta.org

Policy-Driven Mobile Agents for Context-Aware Service in Next Generation Networks

Kun Yang and Alex Galis

University College London, Department of Electronic & Electrical Engineering, Torrington Place, London WC1E 7JE, UK
{kyang,agalis}@ee.ucl.ac.uk

Abstract. In order to bring together the different advantages of various wireless technologies and wired networks, solution from service's perspective is critical. And this kind of integrated services should be context-aware in order to automatically adapt themselves to the changing environment. This paper proposes an all-policy based context-aware service method for next generation networks (NGN). Policies are planned to cover the full context representation through services and the underlying networks. The efficiency of context-aware service provisioning as a result of policy decision is maximized by the introduction of mobile agents. A context-aware service scenario called Super-mother is explored to exemplify this methodology and policy-based context-aware service system architecture. This paper presented part of the work ongoing in European Union IST project CONTEXT.

1 Background and Rationale

One of the main characteristics of next generation networks (NGN) is the convergence of both fixed networks and wireless networks. And this convergence has attracted plenty of researches aiming to bring together the higher speed of wired networks and wider coverage of wireless networks (typically represented by GPRS/UMTS). As the hardware and lower-level structures and protocols of wireless networks get mature, the demands from higher-level applications and services relevant to integrated networks are rapidly growing, especially when wireless LAN (WLAN) technology becomes increasingly popular for providing IP connectivity and 3G is undergoing deployment stage. This paper tends to contribute this literature from the service's perspective by providing context-awareness to the integrated services operating on NGN. We believe the essence of NGN services is the ability of being *context-aware*. Furthermore, the provisioning of context-aware service is facilitated by policy-driven mobile agents.

Bearing a wider scope than *location*, *context* refers to the physical and social situation in which computational devices are embedded [1]. *Context-aware service* (CAS) is more flexible and autonomous so as to respond accordingly to the highly changing computing environments such as location, terminal size, and network features etc without disturbing end user. For example, a cell phone will always vibrate rather than beep during a meeting, if the system can know the location of the cell phone and the meeting schedule. While most of the researches on context-aware computing focus mainly on the human-computer interface (HCI) [1, 2], this paper tends to tackle the context awareness from the perspective of networks, i.e., *network-centric* context-aware services. The networks include both wired IP networks and wireless networks.

To facilitate the provision of context-aware services, apart from an appropriate infrastructure to gather, manage, and disseminate context information to services, the design and development of a context model that takes into account the service and network management is even more important. This context model serves as the basis for CAS system infrastructure. This paper explores the applicability of *policy* for the representing of context in context-aware service and the applicability of mobile agent for the deployment of these context-aware services.

The reason why policies are employed for context modelling is partly because we want to take into consideration the implementation of context-aware services in the underlying networks where policy-based network management (PBNM) is widely regarded as a promising means. Policies are seen as a way to guide the behaviour of a network or distributed system through high-level declarative language in PBNM field, which has been the subject of extensive research as a new network management method over the last decade [3, 4]. As many research works have shown, PBNM technology can relieve network administrator from the burden of configuring every single device manually and it is more flexible since administrator can reconfigure network elements by just giving or changing policies.

On the other hand, in order to provide context aware services, contextual information need to be represented, stored, and maintained. Context information are usually complex, changing, layered and related to each other, which means the use of context information needs complex decision making. Policy-based method fits well to these features of context. Policy-based context modelling can comply very well with the underlying PBNM thanks to the common policy-based schema.

The integration of policy and mobile agents for the automated and flexible management of IP networks has been shown in [5]. This paper tends to take the method one step further by extending it to the provisioning of context-aware service, as depicted in Figure 1.

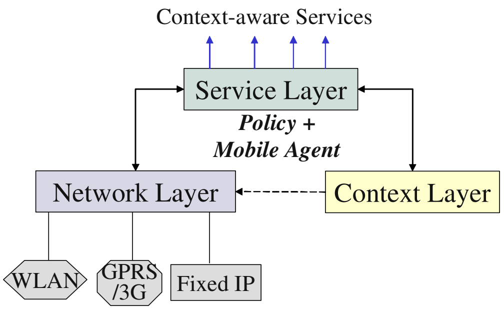


Fig. 1. All-policy CAS Method in NGN Environment

The content of the paper is structured as follows. After the scenario description in Section 2, Section 3 discusses the requirement of context modelling and context-aware service system in term of context definition, context classification and policy specification. Then policy-based context modelling and CAS system architecture are presented in Section 4 and 5 respectively. Before conclusions and future work in Section 7, Section 6 presents the implementation of scenario mainly by means of workflow.

2 Scenario Description

A typical network-centric context-aware service called *TEANU* (Transparent Enterprise Access for Nomadic User) is described, which represents a typical service in NGN.

This service tends to provide a means for nomadic user to maintain the *secure* access to her enterprise network *transparently* after the user has registered for context-aware services from context-aware service provider.

Consider Katherine, a middle class graphic designer with 3 kids. Katherine works from home a few days a week, using her home network that is connected to the office network. On the due day of the project she was working on in the last few weeks, Peter, her 9 years old son, complained that he does not feel well, and his situation degraded to the point that she had to take him to the local public hospital.

Waiting in the unavoidable lines she tries to send her work to the office, using her laptop and a cellular modem. Unfortunately, the bandwidth of the cellular network is insufficient, and transferring the 10GB file will take about 7 hours, far passing the deadline.

However, luckily enough, she had subscribed to this new *TEANU* service that detected bandwidth problem and tried to resolve them. In this scenario the *TEANU* service found out that there was an appropriate wireless LAN (WLAN) access point in the university, and as such it dynamically switched the network connection to this faster wireless LAN. At the same time, a secure tunnel was established automatically between the hospital and her office network. Katherine was then connected to the local network so as to deliver the file faster. She noticed happily the significantly increased file transferring process, and her work was submitted successfully before the deadline. The whole scenario is called *super-mother*. This scenario will be used throughout the paper for exemplifying this all-policy based context-aware service over NGN.

3 Requirement Analysis

Due to the heavily overloaded use of the term *context* with a wide variety of meanings, a clarification of context definition in the context of this paper is the most important requirement for further context modelling and management. The context also needs to be classified so as to guide the design and implementation of context classes in context information model.

3.1 Context Definition

In the work firstly introducing the term *context-awareness* [1], B. Schilit *et al.* refer to context as location, identities of nearby people and objects, and changes to those objects. They claim that the important aspects of context are: *where you are, who you are with, and what resources are nearby*. Dey *et al.* give their definition of context as follow: "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves." [2]. There are plenty of other definitions of context which more or less share the same meaning.

We feel Dey's context definition shouldn't limit the entities to just these that exist only *between user and application*. This constraint, as continuously implied by Dey's later work and most of other context-related work carried out in this field, indicates the origination of context (and context-awareness): Human-Computer Interface (HCI). This paper extends the scope of context research into networks, especially NGN by considering *network-centric context-awareness*. The context definition in this paper is as follow:

Context is any information, obtained either explicitly or implicitly, that can be used to characterize one certain aspect of an entity that involves in a ***specific*** *application or* ***network service****. An entity can be a physical object such as a person, a place, a router, a 3G network gateway, a physical link, or a virtual object such as IPsec tunnel, SNMP agent.*

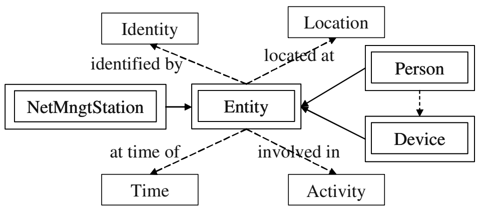
Please note that the collection of context can be done explicitly or implicitly, and an entity can be any network element, wired or wireless.

3.2 Classification of Context

A classification of context types will help application designers decide the most likely pieces of context that will be useful in their applications. Previous definitions of context seed our development of context types. Schilit *et al.* [1] list the important aspects of context as where you are, who you are with and what resources are nearby. Schilit *et al.* didn't consider time this critical factor. Ryan *et al.* suggest context types of location, environment, identity and time [2].

In this paper, sharing the same idea as proposed by Dey [6], we use *location*, *identity*, *time*, and *activity* as basic context types for characterizing the situation of a particular context entity, as depicted in Figure 2.

This context classification clearly answers the questions of who (*Identity*), where (*Location*), when (*Time*), and what (*Activity*) for a specific context entity (*ContextEntity*). This contributes to the horizontal classification of context information. The vertical classification of context information is about context entity itself, which can be categorized as: *persons* including end users with different roles, service provider administrator, network administrator, or *devices* such as 3G mobile handsets, fixed IP routers, WLAN access point, etc, or virtual entities like *network management station*. Furthermore, an object-oriented design of these context types is also carried out in this paper, which is absent in Dey's work.



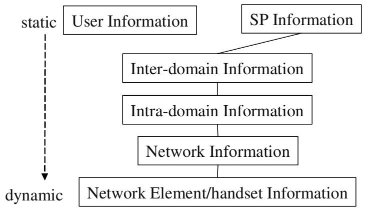


Fig. 2. Classification of Context **Fig. 3.** Network-centric Context Information

As far as network-related context information is concerned, its categories follow the logically hierarchical structure of network as depicted in Figure 3. The higher the context information sits the more static they are.

3.3 Policy Specification

Here policy specification language is rather informal in the sense that it is a kind of high-level English-like declarative language used by administrator to add and update management policies. Policies serve as the containers of context information. As suggested by the Internet Engineering Task Force (IETF) Policy Framework Working Group [4], which has been investigating policies as a means for managing IP-based networks for many years, policies take the following rule-based format:

```
IF {condition(s)} THEN {action(s)}
```

It means *action(s)* is/are taken if the corresponding *condition(s)* is/are true. Policy condition can be in both disjunctive normal form (DNF, an ORed set of AND conditions) or conjunctive normal form (CNF, and ANDed set of OR conditions).

A typical context-aware service scenario can be represented by the following policy, which forces the mobile handset only vibrates rather than beeping during meeting time.

```
IF (location == meetingRoom) and (time within
meetingSchedule) THEN MobileVibratingOnly
```

By this means, any policies related to the CAS can be defined. The execution of these policies eventually triggers the action of classes in policy information model, which are further wrapped by mobile agents travelling around networks.

These rule-based policies are further represented by XML (eXtensible Mark-up Language) due to XML's built-in syntax check and its portability across the heterogeneous platforms.

4 Policy-Based Context Modelling

Context modelling addresses in this paper the issues of how to represent the contextual information in a way that can help bridge the gap between application that uses context information and the deployment of context-awareness. Particularly, an object-oriented (OO) information model should be adopted.

The pioneering work in this area was carried out by Schilit in his PhD thesis [7]. The model of context used in this work was extremely simple, with context information being maintained by a set of environment variables. The most relevant work concerning context modelling is the work lately carried out by K. Henricksen *et* *al.* [8]. This model overcomes problems associated with previous context models such as their lack of formality and generality. As its many precedents, contextual information related to network and network management is not fairly taken into account. Another shortcoming of this work is that all its discussion is on the graphical model while leaving the object-oriented context model unmentioned; therefore a big gap between their context modelling and real implementation of context-awareness is still left unfilled. This is probably due to the lack of a clear picture of the context-aware system that will use these contexts.

A policy-based context information model is designed in this paper based on the IETF PCIM (Policy Core Information Model) and its extensions [9], which are followed by most of the work carried out in policy-based management field.

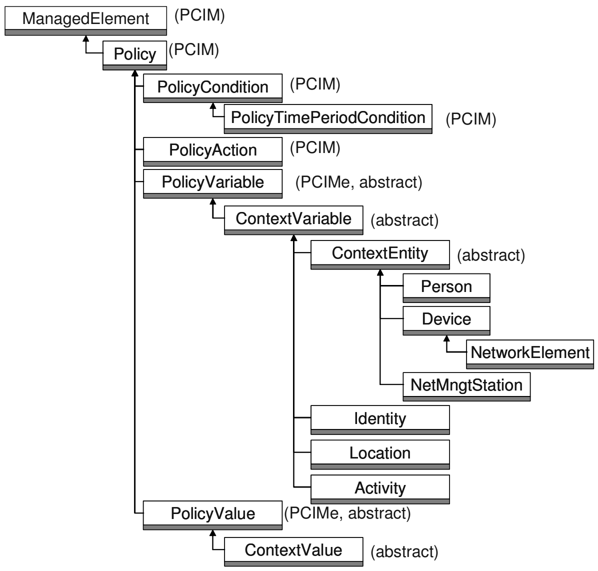


Fig. 4. Class Inheritance Hierarchy in Policy-based Context Information Model

Figure 4 depicts part of the inheritance hierarchy of our information model representing context in network-centric CAS. It also indicates its relationships to IETF PCIM and PCIM extension (simplified as PCIMe in Figure 4). Some of the actions are not directly modelled due to the space limitation. Please note that apart from the policy condition and policy action, context information rooting from *ContextVariable* is mainly reflected in this figure.

Context information is expressed in the policy conditions in the form of elementary Boolean expressions of the form: <*Context Variable*> *MATCH* <*Context Value*>. The relationship "MATCH", which is implicit in the model, is interpreted based on the context variable and the context value. Therefore, the modeling of context information starts from *ContextVariable* and *ContextValue* which inherit two abstract classes available in PCIMe: *PolicyVariable* and *PolicyValue* respectively.

As discussed in classification of context, every context entity should at least be described by four basic features, i.e., identity, location, activity and time. The first three of them are modelled in this class hierarchy as shown in Figure 4. The time feature of context can be directly expressed by *PolicyTimePeriodCondition* which is available as a direct child of *PolicyCondition* in the IETF PCIM.

Now let's have a look at the modelling of network-related context information as described earlier in Figure 3. Since the final fulfilment of network-centric context-aware service is on network elements (including mobile handsets), network elements play a vital part in CAS providing. As such, *NetworkElement* class is defined explicitly as a child of *Device* class. Network information and intra-/inter-domain information are usually maintained by network management stations at different levels or domains. Therefore, network management station (*NetMngtStation*) is defined as a kind of entity that has all the four basic context types. User and Service Provider information can be easily presented by the *Person* class which is actually the child of *ContextEntity* class therefore no separate class for this purpose is needed.

ContextValue is used for modeling context values and constants used in context-related policy conditions. Many extensions of the abstract class *PolicyValue*, as already defined in IETF PCIMe, provide a list of values for basic network attributes which are used directly for context purpose.

5 CAS System Architecture Based on Policy and Mobile Agents

Mobile agent based CAS system architecture, as depicted in Figure 5, has been designed, whose components are organized in term of PBM structure as proposed by IETF Policy Framework Group, which gains wider popularity.

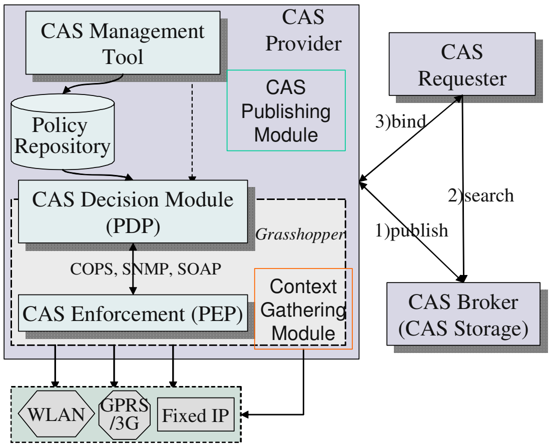


Fig. 5. CAS System Architecture based on PBM and Mobile Agents

As illustrated in Figure 5 from top down, the PBM system for CAS management mainly includes four components: policy management tool, policy repository, Policy Decision Point (PDP) and Policy Enforcement Point (PEP). Policy management tool serves as a policy creation environment for the administrator to define/edit/view policies in a high-level declarative language. Depending on the login name, CAS Management Tool also provides management GUI for network management (mainly

IP VPN) and context information management. After validation, new or updated policies are translated into a kind of object oriented representation or so-called information objects and stored in the policy repository. The policy repository is used for the storage of policies in the form of LDAP directory. Once the new or updated policy is stored, signalling information is sent to the corresponding PDP, which then retrieves the policy and enforces it on PEP.

After a CAS is mature for deployment, CAS provider can publish it to CAS storage, which can be owned by brokers, so that it can be searched for by any CAS requester like Katharine.

Mobile agent technology is explored in this architecture as an enabling tool for flexible service delivery. Mobile agent platform is based on the *Grasshopper*. Grasshopper [10] has been designed in conformance with the Object Management Group's Mobile Agent System Interoperability Facility (*MASIF*). Furthermore, it is a commercial product with extensive documentation and future development under way. However it also provides the version for non-profitable use. In this paper, the mobile agents are fully guided by the policies. One big advantage with mobile agents is that functions can be dynamically provisioned *on the fly*. This feature matches the requirement of mobile user or the features of wireless networks very well.

6 Implementation of *Super-mother*

The test-bed for exemplifying the context-aware integrated service *TEANU* scenario as described earlier in this paper is depicted in Figure 6. The experiment is to simulate the scenario which begins from the point when Katharine enters into the hospital where WLAN is available and ends at the point when the data from the Katharine's laptop begins to transport through an IPsec tunnel between hospital WLAN Access Point and ingress router of her office network.

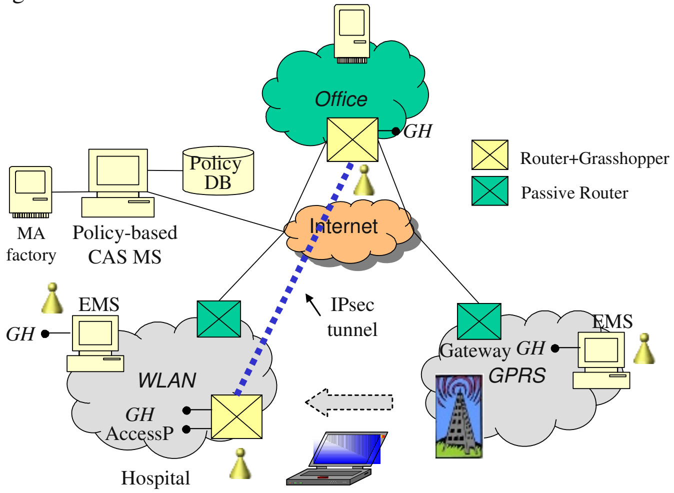


Fig. 6. Test-bed for TEANU Super-mother

Policy-based CAS Management Station (MS), belonging to the Context-Aware Service Provider (CASP), is used by service administrator to management context information, services and networks. EMS (Element Management Station) is introduced for the management of *ABLE* execution environment, access point, etc. Therefore a two-tiered management infrastructure is built up. *MA factory* in Figure 6 serves as the repository of mobile agent classes.

The workflow of this *super-mother* scenario is as follow:

1) When customer (Katharine) subscribes, she should provide CASP with her WLAN card MAC address so as to get access to Access Point (AP).

2) CASP sends this MAC address to all APs that involve in this CAS so that AP can sense the arrival of this specific WLAN card (representing customer himself) and report it to the CASP. These APs are usually given by customer during service subscription according to her selection of possible venues where this service is likely to be applicable. To send customer's MAC address to APs in advance can protect the CASP from being frequently disturbed by unregistered WLAN users.

3) Location sensor (AP in this case) serving as PEP sends a signal to CASP if Katharine enters into the covered scope of AP.

4) Based on the policies in CAS management system and all the relevant context information (such as Katharine's ID, role in the company, etc), CAS Decision Module decides policies to be sent to WLAN domain, and initiate the corresponding mobile agent existing in the MA factory which sends the policy to EMS because AP is not policy-sensitive.

5) After policies arrive at the EMS, EMS begins to configure the access control table of AP to allow Katharine to get access to the AP. Then Katharine's uploading traffic will be automatically switched from GPRS connection to WLAN connection.

6) When the first packet from Katharine's laptop arrives at the AP, its destination address will be checked by Address Checking daemon (a stationary agent) running on the machine where AP is installed. This daemon can find that the packet is to be delivered to her office. Based on the policy that if the packets from Katharine are destined to her office then secure tunnel should be set up for this transferring, IP VPN is going to be set up between hospital and her office. To make the customer side as slim as possible (or without disturbing the user by installing any supporting software such as *Grasshopper* platform or IP VPN software), the secure tunnel is set up between two routers rather than between enterprise router and customer's laptop. Here in this scenario, the machine on which AP is installed also serves as a router.

7) The Address Checking daemon then reports this VPN setup requirement to CAS Management Station (via EMS), which, after further decision making, sends mobile agent to Access Point machine which has *Grasshopper* platform preinstalled to set up one end of IP VPN tunnel in WLAN domain. After this, the mobile agent travels from AP machine all way to ingress router of her office to set up the other end of the tunnel. Then this mobile agent travels back to the management station to report the result. At the same time, customer traffic goes through this secure tunnel.

When uploading is finished (this can be sensed by significant drop of traffic), the TEANU service can switch itself to another level of services to save money.

7 Conclusions and Future Work

This paper provides context-aware service over NGN using the integration of policy-based management and mobile agent technology. This CAS can adapt itself to the change of complex environment dynamically according to the policies resulting from customer-provider contract (or so-called SLA, Service Level Agreement) and rules cast by network administrator for proper use of the networks. The CAS deployment is fully based on mobile agents. Based on the context model and CAS system architecture, a concrete context-aware service scenario *Super-mother* demonstrated the applicability of all-policy method for context-aware services over NGN.

Further refinement of policy-based context model, especially on introducing more WLAN and 3G specific actions, is the main future work. And these actions will be carried out by mobile agents.

Acknowledgements

Part of the work presented in this paper describes part of the work undertaken in the context of the EU IST project CONTEXT (IST-2001-38142-CONTEXT) [11]. The IST programme is partially funded by the Commission of the European Union.

References

[1] B. Schilit, M. Theimer. "Disseminating Active Map Information to Mobile Hosts". *IEEE Network*, 8(5): 22-32, 1994.
[2] A. K. Dey and G. D. Abowd. "Towards a better understanding of context and contextawareness". Proceedings of Workshop on the What, Who, Where, When and How of Context-Awareness, April, 2000.
[3] M. Sloman. "Policy Driven Management For Distributed Systems," *Journal of Network and System Management*, vol. 2, no. 4, Dec. 1994, pp. 333–60.
[4] IETF Policy Framework Working Group web page: http://www.ietf.org/html.charters/policy-charter.html
[5] K. Yang, A. Galis, T. Mota and S. Gouveris. "Automated Management of IP Networks through Policy and Mobile agents". *Proc. of 4th Int. Workshop on Mobile Agents for Telecommunication Applications*: 249-258, Barcelona, 2002.
[6] A. K. Dey. "Providing Architectural Support for Building Context-Aware Applications". PhD thesis, Georgia Institute of Technology, November 2000.
[7] B.N., Schilit. *A Context-Aware System Architecture for Mobile Distributed Computing*, PhD thesis, 1995.
[8] Karen Henricksen, et al. "Modeling Context Information in Pervasivev Computing System". *Proceedings of Pervasive 2002*, LNCS 2414, pp. 167-160, 2002.
[9] B. Moore. "Policy Core Information Model Extensions". IETF-RFC 3460, IETF Policy Framework Working Group, Jan. 2003.
[10] Grasshopper website: http://www.grasshopper.de, 2002
[11] European Union IST Project CONTEXT (IST-2001-38142-CONTEXT) web site: http://context.upc.es/

Merging Virtual and Real Worlds in an Agent-Oriented Collaborative Environment

Michael Zapf and Rolf Reinema

Fraunhofer Institute for Secure Telecooperation (SIT), Darmstadt, Germany

Abstract. Collaborative teamwork is an uprising technology of this decade. With projects spreading around the world, joining partners from distant regions, the idea of distributed teamwork gains importance. While most industrial solutions only focus on a specific set of tools to enhance the communication, an effective teamwork requires the establishment of a real work context. However, the real work environments are lacking influence on the virtual environments. We will show how this can be solved, using an agent-oriented collaboration platform.

1 Introduction

The UNITE platform (Ubiquitous and Integrated Teamwork Environment) [9,8] has been developed as a 2-year IST project from January 2001 to December 2002. During this time, a collaboration environment has been developed which has the following characteristics:

- integrative approach: attempting to incorporate existing solutions into one common frame.
- agent-oriented approach: modelling users and teams by autonomous software agents.

The UNITE project addresses the need of user to be provided with a uniform teamwork environment, whereever they log on to the system. Such environments should allow team members to switch between different projects, make it easy for them to preserve and restore the work context of any given project, allows them to reserve and tailor physical workplaces as team members move and change places, and be sufficiently secured. A collaborative workplace should take care that a uniform and ubiquitous view is presented to all team members regardless of their physical location. Mobile and location-independent collaborative e-Work has been identified as one of the key topics for future research and technological development in the area of New Working Environments, to be supported by IST programs in the 6th EU framework programme [1].

In this article, we want to demonstrate the interactions with the real environment using UNITE as the provider of a virtual teamwork environment. Regarding the principles, other environments may be chosen as well.

This article is structured as follows. In section 2, we will give a short introduction on the basic ideas of the UNITE platform, especially commenting on the underlying agent architecture. Section 3 will present the setting which we want to cover with our new approach. Section 4 demonstrates the extension of the existing architecture and gives a short discussion about security implications of this approach. Finally, section 5 sums up this article and presents an outlook on the future development.

2 The Architecture of UNITE

2.1 Design Rationale

As described in [8], one of the objectives of the UNITE project is to define the *co-operative workplace* as a new working paradigm. People working in teams may participate in different projects. They usually organize their way of working by filtering E-Mail messages, putting them in different mail folders, using various document repositories and so on. Behind this organization strategy lies the idea of sets of resources that are associated to a project. Conceptually, each project has a virtual workspace of its own (called *context*), and anyone who is a member of several projects would go from one virtual workspace to another as he or she goes to work from one project to another.

The UNITE system aims at integrating various different tools in one intuitive user interface, while providing the user with a single sign-on facility and a set of services, representing the work context. One important feature to mention is that UNITE attempts to integrate existing services, not to recreate them. For example, to provide a chat tool and whiteboard, the Lotus *Sametime* system was integrated. On the other hand, other collaboration systems could equally be integrated, while the user does not necessarily notice which service is in use at one team or another.

2.2 UNITE as a Multi-agent Environment

We give a brief description of the existing UNITE architecture. Some further explanations may also be found on the associated web page [9].

The UNITE system uses the AMETAS agent system as an implementation base. AMETAS (Asynchronous Message Transfer Agent System) [5] is a platform for the development of autonomous and mobile agents. AMETAS, formerly developed at the University of Frankfurt, Germany, is now commercially available from the company *accsis GmbH*. Further information about the agent system may be found in [7,6,8]. AMETAS provides places as the execution environment for agents and agent migration as a communications mechanism between them.

The distributed nature of the UNITE infrastructure supports a wide variety of usage and organisational scenarios. In UNITE, we apply autonomous agents not only for entity modelling, but also for handling the distribution of the system. As suggested by the underlying AMETAS infrastructure, UNITE defines two classes of components, the *agents* and the *services*. Agents are intended to carry out the intended workflow to accomplish certain tasks, while service provide an interface to the external world [8].

A virtual team is formed temporarily out of members of several separate organisations. Each participating organisation remains independent, and may participate in multiple virtual teams. Besides participating in virtual teams each organisation still has its private projects and infrastructure. This separation may most easily be provided by introducing several kinds of AMETAS places.

Places. On *homebases*, users and their personal tools, services, and data are represented. A *teamplace* represents one particular team; distinct teams use distinct teamplaces. Thus, the independence of places is maintained within the UNITE platform.

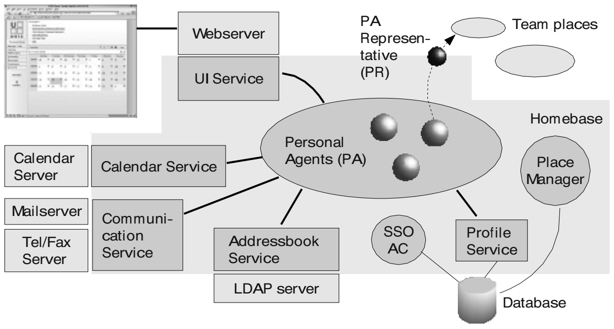


Fig. 1. Homebase structure

Homebases are independent of teamplaces, and teamplaces are independent of each other.

Agents. In UNITE, the following agents are defined:

- *Personal Agent*: runs on homebases, used for authenticating the user and processing tasks according to the user's profile;
- *Team Agent*: runs on teamplaces, used for meeting management and negotiation of communication preferences and capabilities.

The Personal Agent (PA) is actually not a mobile agent; rather, it sends a so-called *representative* (short: PR) to the team which the user is currently working in. The PA continuously runs even when the user is not logged in. In that case, it may still react on special situations in the platform on behalf of the user, for instance, by forwarding events to him or her as a short message on the mobile phone.

For information exchange between the places, *messengers* are used. These are mobile agents which deliver the given message to the specified Place User at the remote place. Messengers are protected from outside influence by the encryption applied by the AMETAS security system.

Services. There is a list of services which have been implemented in the UNITE platform:

- *Addressbook Service*, *Calendar Service*, *Communication Service*: interfacing to external resources like LDAP, calendar server, or collaboration system (like Lotus *Sametime*), respectively;
- *Profile Service*, *Team Agent database service*: providing persistency for the PAs and TAs by storing and retrieving data from a MySQL database;
- *Place Manager*: infrastructural component; cares for starting up all components and allows administrative actions;
- *UI Service*: interfacing to the web server (specifically, Tomcat servlet engine)

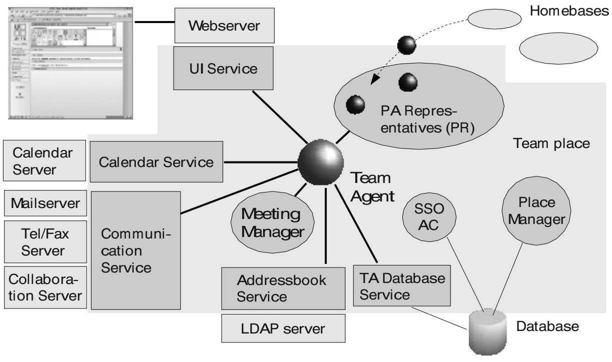


Fig. 2. Teamplace structure

Homebase At the homebase, Personal Agents are maintained of all users who take this homebase as their UNITE login gate (Figure 1). The Personal Agents are launched automatically when starting the place, and they continue running after the user logged out again.

The *UI service* gets the user actions, translates them to messages, and directs them to the respective Personal Agent. Vice versa, messages coming back from the Personal Agent as a result of the action are translated into a suitable output frame to be displayed by the user's browser. Auxiliary services may be found at the homebase which provide access to features like the user's private calendar, or the own document repository.

Team place Similar to homebases hosting the Personal Agents, a teamplace hosts a Team agent (Figure 2). When a user enters the team context managed by this teamplace, the PA representative migrates to this place, carrying the personal profile part related to this project with it. This profile will then be used for handling communication negotiations, according to personal preferences.

The Team agent also handles two kinds of *virtual meetings*:

- *instant meetings*: Two or more users start a collaboration session instantly.
- *scheduled meetings*: A meeting initiator creates a meeting to take place in the future. In this scenario, the Team agent is responsible (via its Meeting manager) to create the respective entries in the team calendar and to notify all persons to be invited.

On the team place, similar auxiliary services may be found which allow to access the team address book, the team calender and so on. Again, there is a UI service which provides an interface between the agent world and the servlet engine used to create the user interface.

Generally, a homebase represents a set of users and can thus be used to represent an organisation, a unit within an organisation, or just an arbitrary collection of people. A teamplace, being an independent entity, can be run by an organisation participating in the respective project or by an independent third party, a service provider.

3 Sensing the Real World

The previous sections described the situation as addressed by UNITE and many other collaboration environments. However, all of these approaches did not yet take into account the real workspace.

3.1 Room and Facility Management

Room management, or more general *facility management* refers to the control of real-world rooms, buildings or installations by means of electronic devices. New approaches try to make buildings be used for working as well as for living, just adaptable by the user's preferences. Office design will be greatly influenced; other than the fixed allocation of space, movable walls and different layouts will introduce new degrees of freedom to the design of the workplace.

This flexibility also raises a new demand of management of rooms and devices in this office landscape. Unlike today, it might not be adequate to find a person by a room number – because the room does not exist any more. Bus systems allow for moving devices without the need for reconfiguration, but again the question remains to locate these devices once they are not bound to a specific location. The *RoomComputer* [10] is a PC-like device which allows to introduce the management of devices and rooms in a flexible way, interfacing to different field buses, using a web interface. One of the salient advantages is the possibility to distribute an array of RoomComputers in an office structure, each one managing some part of the building.

The facility management system (specifically, the backend part) may consist of a database system, together with provisions for sensing the presence of specially equipped resources. These sensors may detect and read RFID tags stuck on devices. With Office ID cards (OID), users may inform the system about their location, for instance, to get a customized working environment.

3.2 Virtual Meetings and Real Places

Current collaboration tools, including UNITE at the current state, usually assume meetings to be purely virtual. Handling of virtual meetings focuses on providing the technical tools (like collaboration tools) in terms of infrastructural support. However, the focus loses the aspect of people *actually requiring to gather somewhere*.

For tools currently in use, gathering at some common location is of lower interest because there is the assumption that either the user may take their devices with them, using wireless phones or laptops, or to be rechable at a well-defined location (office room). This proved to be enough for the set of currently available tool which address working in a virtual environment, accessing common services deployed in the network.

Currently, the usage of collaboration tools is restricted to a usage of server-based tools with small-footprint client devices (telephones, laptops, or PDAs). Future communication devices may show different handling. Considering sophisticated projection devices like 3-dimensional projection (up to CAVEs [2]), or SmartBoards [3], it becomes clear that the physical location of a person *does* matter because it is not appropriate for each co-worker to be equipped with these features. Moreover, there is a trend of using non-territorial devices, last but not least for the sake of cost savings.

Video conferencing, as another example, is a very bandwidth-comsuming action which is generally handled by small window sizes or poor quality. On the other hand, you can also use a video connection between two peers only, which nevertheless allows many more users to participate at the same time if they are collocated in the room. Here, a much better quality can be achieved. This may lead to communication parameters which depend on the expected quality of service (QoS) of the communication.

3.3 The Roomplace

The current UNITE architecture as shown in section 2.2 assumes this traditional setting of users working in virtual teams. For incorporating the management of facility, a new structure must be introduced which we call *roomplace*.

The roomplace is intended to provide services interfacing to rooms and facilities management systems. Moreover, there must be an interface to the management of devices in this new environment. This will enable applications to locate devices which are required for communication purposes.

This room and device management scenario provides yet another good application scenario for mobile autonomous software agents. By using agents, the implementation is well-structured and encapsulated in self-contained entities.

Flexibility of the agent architecture. The inclusion of room and device management merely requires the addition of one type of infrastructural component. Similar to the homebases and teamplaces, the roomplace will host at least one *Room agent* which cares for the access of traditional UNITE components to the management system. Several Room agents may be installed to represent real-world hierarchies like building parts or levels.

Distributed structure. Moreover, the application of mobility allows for a distributed structure. It is not required to design complex client-server protocols which cause a lot of overhead for planning and allocating conference venues. As a distributed teamwork environment, the UNITE system already adopted the usage of mobile agents to transport information between the places.

Natural extension of the virtuality. The abstraction of devices and rooms by agents is an analogy of the representation of users and teams by agents inside the system. On the implementation level, the Room agent *abstracts* from the details of the resouce management system and hereby allows to flexibly integrate different systems in a uniform way.

4 The System in Action

We will now discuss two scenarios which employ the room management system, in conjunction with the team collaboration features of UNITE. There are some prerequisites, of course:

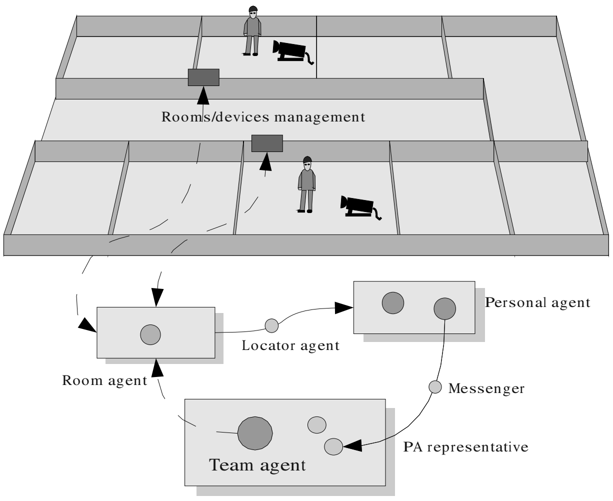


Fig. 3. Room management

Users must have a means of entering information about their current location into the system. This should be as simple as possible, with minimal actions from the user. Here, two approached could be useful:

- Office ID cards, to be used with a card reader when entering the room.
- RFID (Radio Frequency ID) tags/cards which allow for a passive detection of the user [4].

All these users must have existing accounts at the teamwork environment. Devices which need to be used for collaboration and which shall be managed by the infrastructure must also be registered in a facility management system interfacing to the collaboration platform.

4.1 Instant Meetings and Moving Persons

Consider the following scenario:

- Two persons (assumed to be in the same facility) intend to have an instant meeting. In the Figure 3, we see two people and a device (e.g. a projection device) in each of the rooms.
- One of the persons selects the communication feature which shall connect to to the second person.

In the current UNITE version, we do not have a information about the real world, so we expect the persons to be either in the expected rooms, or to have mobile devices.

Detecting the location The required information about the location of the persons may be retrieved by an integrated room management system. Assume that persons are locatable by means of office ID cards or RFID tags, the room management system knows the current locations of the users.

This information may be queried by roomplaces which are in contact with these room management systems. This is achieved by employing interfacing components, converting the agent system-specific communication processes into the respective API calls to the management system. Hence, *Room agents* playing an analogous role of the Team agent or Personal agents for the devices and rooms, query this information from the interfacing service and need to inform the appropriate components of the platform. In this case, we suggest to pass on this location information to the Personal agents (PAs) on the homebases. Unlike the Team agents, Personal agents are not bound to processing project-related information, and the physical location of the user is a personal, not a project-related issue. Moreover, the Room agent simply *does not know and need not know* in which project the user is currently active.

Finding the user's homebase The next task is to find out how to reach the Personal agent of the detected user. We cannot assume that this information is provided with the identification of the user by the RFID tag or the OID card.

In the case that the user's identification device was issued by the facility where he or she is currently working, the solution would be to allow the detection system to download parts of the user's profile stored in some database connected to the detection system, containing the information about the user's homebase. (The profile management system must be adequately configurable.)

Considering visitors from other UNITE-enabled locations, two ways of handling can be envisaged: Either the institution has to issue temporary tags, or the visitors have to be announced to the system manually. The second option could look like asking the person to authenticate to the system in the room intended for use. Having got the information about the user's homebase, the Room agent now emits a messenger which travels to this homebase. The homebase itself cares for the correct mapping of the user identity to the appropriate Personal agent.

Propagating the location information to the teamplace Unlike the Room agents, the Personal agents know about the activity of the persons. Thus, the PAs can contact the current teamplace, or, specifically, the associated Personal agent representatives (PRs) which are running at that teamplace. This is again performed using messenger agents.

The PRs require the information for the negotiation strategy to set up a collaboration. For example, according to the user's preferences, the location information may influence the selection of collaboration tools. Once the PRs have decided on a collaboration scenario, the services may be started. Here, the Team agent may need to get in contact with the respective roomplaces which manage the devices to be used.

4.2 Scheduled Meetings and Moving Devices

We encounter a different setting when we think about the planning of virtual meeting, together with the rooms and devices management. In spontaneous meetings, as described above, the user location is instantly detectable, allowing to negotiate the communication tools in dependency to the location. Scheduled meetings refer to some future setting.

The tasks for the set-up of a scheduled meeting may be summarized as follows:

- Define the set of users to participate.
- Define the collaboration tools to be used.
- Check the availability of devices and locations at the planned time.

Common collaboration tools allow for setting the set of users to be invited to a meeting. The definition of collaboration tools, however, strongly depends on the availability of devices and locations. Using simple webcams and microphones, this would not be an issue, but for our considerations, we again think about non-territorial devices.

Therefore, we have a inter-relationship between *requested* tools and *available* tools, which in turn may influence the setting of the meeting as such. For example, if the projector room is not available at the requested time, another date needs to be found. Another way to cope with such a problem would be to *decouple* devices and rooms if possible: *If the room is not available, then what about moving the device to another room?* This introduces another degree of freedom for the management of collaboration meetings. Here, we suggest to give devices RFID tags to detect their physical locations, similar to the above scenario. Using more "intelligent" devices, they could also announce their location actively to the management system.

Find the resource. If a device is detected to be present in a room, the detection system causes a state change in the management system. For a proper resource management procedure, these informations must be accessible to the *Meeting Manager* of the teamplace. When the meeting initiator selects some communication tools for the meeting, the Meeting Manager will need to look up the location of these devices, reserve them to be available for the intended meeting.

Check at remote places. In order to get an allocation of resources for a meeting, it could be required to send an agent to remote roomplaces. Here, the agent could do the same check as done locally in order to find out about available resources for the meeting. Note that this will surely be subject to all partners agreeing to cross-organizational facility management.

Negotiation. In the case that every resource is available, the Meeting Manager will present the result to the meeting initiator. The initiator may then trigger the sending of invitations to the participants. This feature is already available in UNITE, as with most other collaboration systems.

If not all resources will be available, the meeting initiator should be presented the possible choices for alternative dates and/or resources. For example, if the desired device will not be available, the system could suggest to use an equally useful, available device for the meeting.

4.3 Security and Privacy Considerations

The ubiquitous detection of personal presence and locations raises questions about security, especially privacy. The system as proposed here is capable of finding out whether a person is collocated with some device, and whether two persons are currently collocated in the same room.

This may not only degrade the user acceptance, it might also violate labor laws if it is found to be useful for monitoring the users behavior (e.g. how much time does the user spend away from his desk). Although this is a problem which is already founded in the application of RFID tags or office ID cards, the combination with a collaborative system could lead to the perception of a monitoring system where anybody can watch any other participant at any time. The system should therefore implement reasonable policies which are certified to respect these privacy side-conditions.

Another possible set of attacks concerns the delivery of location and availablility data. With this respect, however, the agent infrastructure should provide adequate means of intrusion detection and tampering or eavesdropping. The management of devices and locations must be protected from outside influences. This is again true for the management system as such – the usage within the teamwork environment does not infer new issues.

Room and resource management crossing organisational borders could prove to be difficult, especially because each organisation claims to exert property rights on their own equipment and rooms – reserving such resources in another company even if involved in the same project will most certainly not be feasible in an easy way.

5 Conclusions and Future Work

In this article, we demonstrated the application of agent-oriented software in the context of virtual teams, with the extension to connect to the real world. Agents do not only allow to represent the real user in a virtual environment, they also help in synchronizing the internal and external states of the system. Using this feature, it is possible to augment the virtual world with information from the real world – the opposite direction of what is known as *augmented reality*.

The UNITE project introduced a context-oriented teamwork environment based on an agent architecture. We found that the decision to employ autonomous and mobile software agents for the design of the components was a good choice because we now have the possibility to easily extend the system and use agent with possibly different resource allocation policies without the need to care about installing the software at all locations within the teamwork environment. The mobile agents just wander to these locations for fulfilling their tasks. It is simple to replace negotiation strategies during the runtime of the system just by replacing the agents.

The architecture described in this article shows many potentials for extensions. The development of this architectural extension is currently work in progress. From earlier projects, we already collected some experience in the application of RFID techniques, hoping to exploit this knowledge for this extension.

Finally, we pointed out that beyond solving technical problems, we need to care about the usage scenarios offered by a system allowing to process the location of people together with collaboration scenarios. Legal issues and the acceptance of the users must be taken into account.

References

1. European Commission: Decision No 1513/2002/EC of the European Parliament and of the council of 27th June 2002 concerning the Sixth Framework Programme of the European Community for research, technological development and demonstration activities, published in the Official Journal on 29.08.2002, http://europa.eu.int/comm/research/fp6/
2. C. Cruz-Neira, D. J. Sandin, T. A. Defanti et al. (1992). The CAVE: audio visual experience automatic virtual environment. Communications of the ACM, 35(6), 64-72.
3. SMART technologies: SMART Board Interactive Whiteboard. http://www.smarttech.com/products/smartboard/
4. International Organization for Standardisation: ISO/IEC JTC 1 / SC17 website, Cards and Personal Identification. http://www.sc17.com
5. accsis GmbH: AMETAS homepage. http://www.ametas.de/
6. M. Zapf: Type-based mediation of Mobile Agents. Proceedings of the International ICSC Congress "Intelligent Systems and Applications ISA 2000", Wollongong, Australia.
7. M. Zapf: Typisierung autonomer Softwareagenten. Dissertation, Goethe-Universität Frankfurt/Main, Germany, 2001. Published by dissertation.de, Berlin 2002. ISBN 3–89825–402–X. Available online via http://www.dissertation.de/
8. M. Zapf, R. Reinema et al.: UNITE – an Agent-Oriented Teamwork Environment. Proceedings of the 4th International Workshop on Mobile Agents for Telecommunication Applications (MAMA 2002), Barcelona, Spain 2002. ISSN 0302–9743.
9. UNITE consortium: The UNITE web pages. http://www.unite-project.org/
10. R. Reinema, H. Thielmann: RoomComputer – Ubiquitous Computing in Cooperative Rooms. In: thema Forschung, 1/2002, pp. 118–125, Technische Universität Darmstadt, Germany. ISSN 1434-7768. See also: http://www.raumcomputer.de.

A Formal Model of Active Contents Based on the Ambient Calculus

Yasuyuki Tahara[1], Nobukazu Yoshioka[1], and Shinichi Honiden[2]

[1] National Institute of Informatics
[2] National Institute of Informatics / The University of Tokyo,
2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, Japan
Tel: +81-3-4212-2513 Fax: +81-3-3556-1916
nobukazu@nii.ac.jp

Abstract. The recent innovation of telecommunication and networking technology is enabling easy and flexible distribution of digital multimedia contents. However, such rapid progress has brought about various problems on intellectual properties and security. We are investigating a technique to solve the problem called active contents based on hierarchical structures of mobile agents. The agents work as wrappers of contents and can easily manage the policies for contents distribution. In this paper, we give a formal model of active contents in order to establish rigorous foundations for the active contents technique, especially the system of the policy control mechanisms. Using the model, we can verify if the behaviors of the active contents satisfy the given policies or not. An example of the redistribution prohibition policy illustrates how the verification works.

1 Introduction

The recent innovation of telecommunication and networking technology is enabling easy and flexible distribution of digital multimedia contents. On-demand contents delivery is now available using the Internet and PCs at home. Even outdoors, we can access contents using PDAs or cellular phones. On the other hand, such rapid progress has brought about various problems including violations of intellectual properties and privacy and security concerns about personal information and payment. We have difficulties in coping with these problems because there are various requirements for these issues that change as time goes by. Such requirements sometimes contradict each other when they are given at the same time and we should coordinate them in this case. However, since the number of the requirements is usually very large, it is difficult to manage them and make the contents distribution work well.

We are investigating a technique to solve the problem called *active contents*. The active contents technique is based on the mobile agent technology and enhances the multimedia contents with autonomous behaviors and policy control mechanisms with the agents. Since the agents work as wrappers of the contents carrying the policies that are requirement specifications for conditions in distributing and using the contents, it is easy to force the policies to access to the contents. The complicated relationships of the various policies are organized by the hierarchical structure of the agents.

In this paper, we give a formal model of active contents in order to establish rigorous foundations for the active contents technique, especially the system of the policy control mechanisms. This model is based on the Ambient Calculus [1, 2], one of the most widely used formal models of mobile agents. In this model, the policies are modeled as requirement specifications represented by logical formulae. On the other hand, the system behaviors are modeled as behavior specifications represented by process expressions. We can verify if the system behaviors satisfy the policies or not by trying to prove formally that the process expressions satisfy the logical formulae or not.

This paper is organized as follows. Section 2 explains the notion of active contents. Section 3 describes the detail of the formal model of active contents. Section 4 illustrates how our model works in verification of possibility of policy violation. Section 5 discusses some related work and Section 6 gives some concluding remarks and future work.

2 Active Contents

The active contents technique is based on the mobile agent technology and enhances the multimedia contents with autonomous behaviors and policy control mechanisms with the agents. Recent multimedia contents distribution environments is creating various problems about intellectual properties and security concerns. A number of policies to consider these problems are appearing corresponding to the problems as follows.

– Policies of contents providers include distribution limitations such as prohibition of redistribution, limitations of information access, and prices.
– Policies of contents consumers include access control for contents harmful to children, reuse policies and price negotiation.

In addition, some of these policies may contradict each other and in this case we need to negotiate and compromise with other parties. The active contents technique makes it easy to manage such situations. An active content is a system of agents that carry the contents to be distributed. We have the following three types of agents (see Figure 1).

A contents agent protects the contents to be distributed. It behaves so that the policies of the contents creator are kept correctly. It also provides the cryptography mechanisms for the contents.

A container agent carries composite contents consisting of one or more contents agents. It behaves so that the policies for the combination of the individual contents given by the creator are kept correctly.

A distribution agent has the responsibility to deliver the container agent carrying the contents to the contents consumer by migrating from the provider's host to the consumer's host. It manages the distribution policy and behaves according to the policy.

These types of agents form a hierarchical structure in which an upper type is carried by the lower one. Contents distribution is represented by migration and transfer of the agents (see Figure 2).

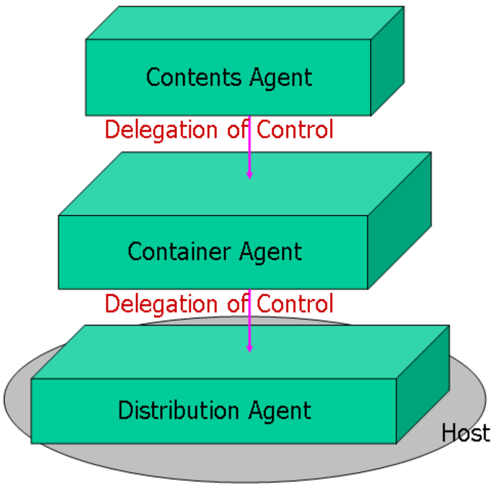


Fig. 1. Structure of Active Contents

3 Formal Model of Active Contents

In this section, we present a formal model of active contents based on the Ambient Calculus. We explain first the Ambient Calculus that is the foundation of the formal model. We also explain how to make use of the formal model in development of active contents systems.

3.1 Ambient Calculus

The Ambient Calculus [1, 2] is a formal model of concurrent computation similar as π-calculus [5]. Its main feature is that it can represent mobile agents by the entity called *ambients* that can also represent hosts. The Ambient Calculus is one of the most popular formal models among the similar ones.

The elementary construct of the Ambient Calculus is *name* representing an identifier of a software module or a communication channel in the same way as the π-calculus. The Ambient Calculus represents system behaviors by reductions of processes composed by the parallel composition operator, sequences of actions ("capabilities"), interprocess communications, etc. For example,

- $\mathbf{0}$: inaction
- $P|Q$: parallel composition of the processes P and Q
- $(x).P$: input a data and call P with assigning the data to the variable x

An *ambient*, the characteristic component of this calculus, takes the following format.

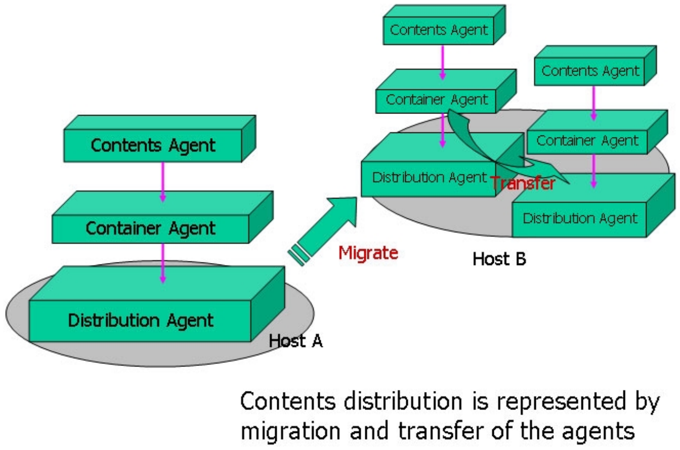


Fig. 2. Representation of Contents Distribution

$n[P]$ where n is a name and P is a process

Thus an ambient is an encapsulated process with a name as the identifier. Since an ambient is also a process, ambients can be nested. A child ambient can go out from or enter its parent ambient. An ambient can be "opened", that is, the name is ripped off and the content process is exposed. A migration of an agent can be represented by the two steps in which the ambient denoting the agent goes out of the ambient denoting the starting location and then enters the third ambient denoting the destination location. The reduction rules for these actions can be described as follows.

- $n[\texttt{in}\, m.P|Q]|m[R] \longrightarrow m[n[P|Q]|R]$
- $m[n[\texttt{out}\, m.P|Q]|R] \longrightarrow n[P|Q]|m[R]$
- $\texttt{open}\, n.P|n[Q] \rightarrow P|Q$

We have a modal logic system for the Ambient Calculus [1]. This logic consists of logical formulae that can use the parallel composition and the ambient construction operators, the temporal modality "sometime" ($\Diamond$) and "always" ($\Box$), the spatial modality "somewhere" (✧) and "everywhere" (⌑) in addition to the usual first-order logic operators. We can define the satisfaction relationship between a process description P and a logical formula Φ expressed as $P \models \Phi$. The system development process use this relationship for verifying if the requirement specifications represented by Φ satisfy the behavior specifications represented by P. In the context of active contents, we treat the contents distribution policies as the requirement specifications.

3.2 A Formal Model of Active Contents Using the Ambient Calculus

We use the Ambient Calculus to give a formal model of active contents. The summary of the model is as follows.

- System behaviors are represented by process descriptions. Especially, an agent is represented by an ambient. The hierarchical structure of the three types of the active contents component is represented by nested ambients.
- A policy is represented by a logical formula the process description is expected to satisfy.

The following figures are examples of system behavior modeling. Figure 3, 4, and 4 shows the modeling of, respectively, a general hierarchical structure, the migration step of the distribution agent, and the step in which the container agent is transfered.

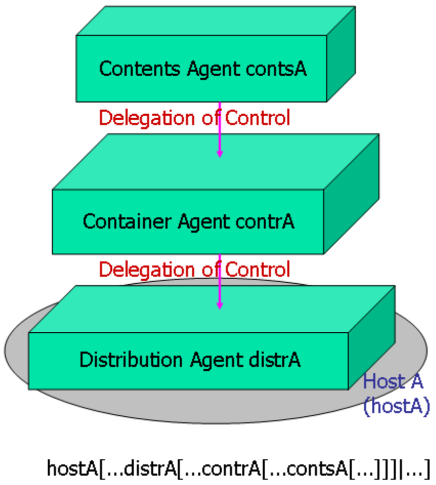
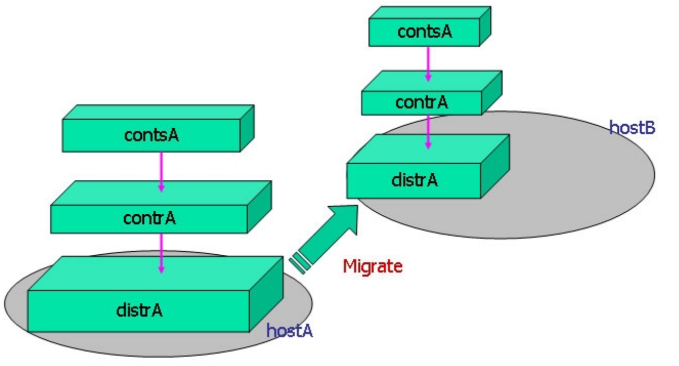


Fig. 3. Modeling of a General Hierarchical Structure

4 Example

In this section, we illustrate how our formalization works as the basis of verification of policy control for active contents using an example of illegal contents redistribution detection.



```
hostA[distrA[out hostA. in hostB...contrA[...contsA[...]]]|Pa]|hostB[Pb]
-> hostA[Pa]|hostB[distrA[...contrA[...contsA[...]]]|Pb]
```

Fig. 4. Modeling of the Migration Step of the Distribution Agent

4.1 Details of the Example

This example treats the following scenario. A contents provider p sells contents c to a buyer b with the policy prohibiting redistribution of c to a third party. p attach c the digital watermark consists of the provider and buyer information. p uses the redistribution detection mechanism that searches the contents attached the watermark including the provider information of p in the network.

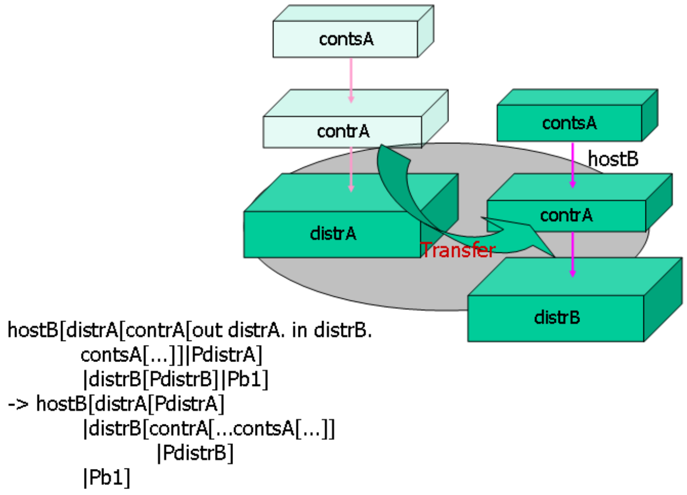
The formal modeling of the policy is the following logical formula.

```
always forall X.
  ((X =\= hostP and X =\= distrP and
    ... and X =\= hostB and somewhere X[somewhere c])
  implies eventually somewhere hostP[
    somewhere knows[redistributed[host[<hostB>]|conts[c]]]])
```

where hostP and hostB represents the host of p and b and X represents an ambient name that is not allowed to contain the contents.

We need the following assumptions in order to formalize the system behaviors.

- We have a host hostDB storing information of all of the host existing in the network. We can suppose that this is like a Web search engine or a directory server.
- wm(c, k, w) denotes the contents in which the watermark w is attached to the raw contents c with the secret key k. deWM(k) denotes the process that can extract c and w from wm(c, k, w). Formally,



```
hostB[distrA[contrA[out distrA. in distrB.
          contsA[...]]|PdistrA]
          |distrB[PdistrB]|Pb1]
-> hostB[distrA[PdistrA]
          |distrB[contrA[...contsA[...]]
                    |PdistrB]
          |Pb1]
```

Fig. 5. Modeling of the Transfer Step

```
wm(c, k, w)|deWM(k) -> conts[c]|waterm[w]
```

For example, we can define

```
wm(c, k, w) := k[conts[c]|waterm[w]]
deWM(k) := open k. 0
```

– `ifEQ` is the function for "if-then-else". In detail, `ifEQ(x, y, p, q) -> p` if `x = y`, `q` otherwise.
– `extrBuyer` is the function extracting the buyer's host name from a watermark.

The process description that is the objective of the verification should include the process realizing the behaviors of the provider's system and the process representing an environment including a possible malicious attacker. In detail, the description is the parallel composition of the following components.

– The behavior specifications of the provider's system `specP` are

```
hostP[distrP[out hostP. in hostB.
      contrP[in distrB.
        contsP[wm(c, k, provider[<hostP>]|buyer[<hostB>])]]]
  | wmFinder[out p. in hostDB. (Host). in Host.
      (deWM(k)|open waterm. (WM).
        ifEQ(extBuyer(WM), Host, 0, detected[out wmFinder.
          out Host. in hostP. open conts. (C).
          knows[redistributed[host[<extrBuyer(WM)>]
                |conts[C]]]]|<Host>))]]
```

representing that
- the distribution agent `distrP` of `p` migrates from `hostP` to `hostB` and transfers the container agent `contrP` to the distribution agent `distrB` of `b` and
- the agent `wmFinder` concurrently moves around the network and searches the contents with the watermark and reports the search results if it finds such contents.

Figure 6 illustrates these behaviors.

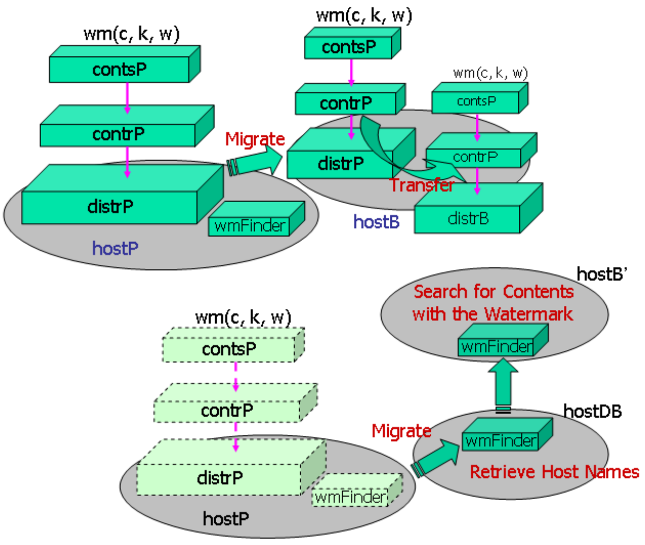


Fig. 6. Behavior Specifications of the Provider's System

– An environment in which the buyer `b` is a possible malicious attacker is modeled as follows.

```
hostB[distrB[open contrP. open contsP. out hostB. in hostB'.
                0]]|hostB'[]
```

where `hostB` is the host of `b` and `hostB'` is the redistribution target. `distrB` takes out the contents out of `contsP` and hands them to `hostB'`. Let this description `evilEnv`.

We demonstrate the following verification procedure (see Figure 7) in which we check if the composition of these two process descriptions `specP|evilEnv` satisfies the logical formula representing the policy.

1. First, we need to investigate the reduction results from the initial process description exhaustively in which `c` appears in some ambient that may not

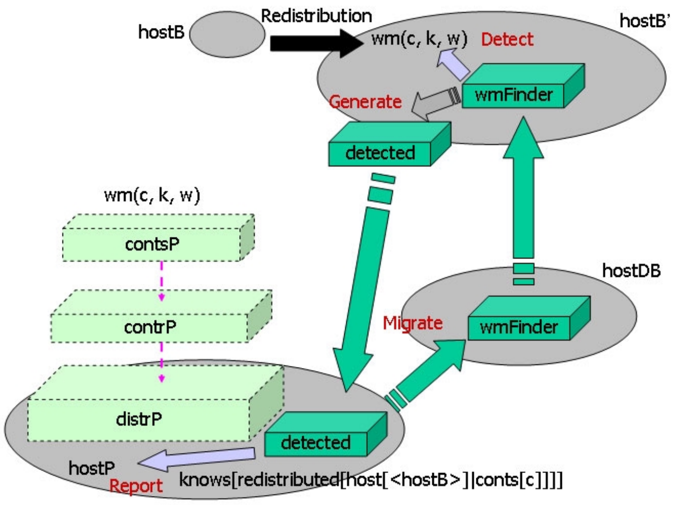


Fig. 7. Example of Verification Procedure

contain the contents. In this example, `hostB'` can contain the contents by receiving them from `hostB`.

2. We investigate further the reduction results from the process states listed up in the previous step and check if the violation of the redistribution prohibition policy is reported in one of the states investigated in this step.

5 Related Work

Recently, hierarchical models of mobile agents are investigated by several researchers. Lime [6] is a programming model that is based on Linda [4] in which a mobile agent carries a private tuple space, migrates between common tuple spaces and communicate with other agents via the common tuple space. The hierarchical structure of the tuple spaces can be used as the programming model of active contents. MobileSpaces [9] is another hierarchical mobile agent framework in which an agent can move into and out of another agent. The author of the paper also proposed a formal model [7] and a multimedia application [8], although policies for contents distribution are not considered. It is interesting to investigate application of these models to active contents.

The notion of Policies in general contexts is also a hot research topic. Several description languages of policies such as Ponder [3] and formal models of policies are proposed. However there are no languages or formal models specific to the application of mobile agents to multimedia contents distribution. We are now planning to incorporate the existing research results of policies into the active contents technique and its formal model proposed in this paper.

6 Conclusions

In this paper, we gave a formal model of active contents in order to establish rigorous foundations for the active contents technique, especially the system of the policy control mechanisms. This model is based on the Ambient Calculus one of the most widely used formal models of mobile agents. In this model, the policies are modeled as requirement specifications represented by logical formulae. On the other hand, the system behaviors are modeled as behavior specifications represented by process expressions. We can verify if the system behaviors satisfy the policies or not by trying to prove formally that the process expressions satisfy the logical formulae or not.

We are investigating the following future work.

- The active contents technique itself is still at an early stage. We consider that our formal model helps to improve the technique in a right direction by enhance the technique without breaking the rigorousness.
- We need to use the existing related research results described in Section 5. For example, it is important to develop a policy description language for active contents by considering the existing languages.
- Formal models are difficult to apply to practical situations in general. We should examine development processes and tools based on our model as a direct practical application of our model similarly as [10].

References

1. L. Cardelli. Mobility and security. In F. L. Bauer and M. C. Mult, editors, *Foundations of Secure Computation*, NATO Science Series: Computers & Systems Sciences, pages 3–37, 2000.
2. L. Cardelli and A. D. Gordon. Mobile ambients. In M. Nivat, editor, *Proc. FoSSaCS'98, LNCS1378*, pages 140–155. Springer, 1998.
3. N. Damianou, N. Duray, E. Lupu, and M. Sloman. The Ponder policy specification language. In *Proc. Policy 2001*, volume 1995 of *LNCS*, pages 17–28. Springer, 2001.
4. D. Gelernter. Generative communication in Linda. *ACM Computing Surveys*, 7(1):80–112, 1985.
5. R. Milner. *Communicating and Mobile Systems: the Pi-Calculus.* Cambridge University Press, 1999.
6. A. L. Murphy, G. P. Picco, and G.-C. Roman. Lime: A middleware for physical and logical mobility. In *Proc. of ICDCS-21*, pages 524–233. IEEE Computer Society, 2001.
7. I. Satoh. A formalism for hierarchical mobile agents. In *Proc. of PDSE 2000*, pages 165–172. IEEE Computer Society, 2000.
8. I. Satoh. A hierarchical model of mobile agents and its multimedia applications. In *Proc. of MMNS 2000*, pages 103–108. IEEE Computer Society, 2000.
9. I. Satoh. Mobilespaces: A framework for building adaptive distributed applications using a hierarchical mobile agent system. In *Proc. of ICDCS 2000*, pages 161–168. IEEE Computer Society, 2000.
10. Y. Tahara, A. Ohsuga, and S. Honiden. Mobile agent security with the IPEditor development tool and the Mobile UNITY language. In *Proc. of Agents 2001*, pages 656–662. ACM Press, 2001.

Introducing MPLS in Mobile Data Networks: An High Performance Framework for QoS-Powered IP Mobility

Francesco Palmieri and Ugo Fiore

Università degli Studi di Napoli Federico II
Centro di Servizi Didattico-Scientifico,
Complesso Universitario di Monte S. Angelo, via Cinthia 45,
80126 Napoli Italia
{fpalmieri,ufiore}@unina.it

Abstract. Because of the evolution of portable computing, and personal communication technologies, mobile Internet connectivity is the fastest growing business in the telecommunications market, playing a vital role in shaping the 21st century communications paradigms. In this scenario, the deployment of innovative wireless data networks, the integration with the Internet and the interworking between different wireless technologies will be challenging objectives for competitive service providers. These factors, combined with the impact that mobile related traffic may have on the fixed infrastructure, and the convergence of mobile and fixed services, drive towards a rationalization of the resource allocation and management procedures and make it urgent to address the node mobility problem from a global, core-level traffic engineering point of view. We propose a framework for the integration of IP mobility and MPLS in the mobile data network focusing on the use of consolidated technology, with no major changes to standardized protocols or devices. Our model that handles wireless IP device mobility by combining local area mobility techniques at the edge and MPLS in the backbone, allows very fast handovers without the need of modifying the IP address, works with any IP version, has a low header overhead (compared to IP-in-IP tunneling), and can get the native traffic engineering and QoS benefits provided by MPLS to continuously adequate the traffic flows in the mobile data network backbone to the dynamically changing traffic requirements.

1 Introduction

In the last decade, mobile data communication and Internet connectivity services have been experiencing an explosive growth to mass-market dimensions. Experience with laptop computers and personal digital assistants (PDAs) has shown that many end users desire their portable equipment to provide essentially the same environment and applications they enjoy at their desks with few compromises thus demonstrating the singular importance of widespread coverage and anywhere/anytime access. Furthermore, with an increasing demand for the various mobile wireless data services such as wireless internet, video on demand (VoD), video telephony, multimedia mail, etc.

through the 2.5G or 3G systems, the number of subscribers to the data services would keep growing at an astonishing pace. Accordingly, mobile-related traffic is forecast to be comparable in volume with that related to fixed networks in a not too distant future. These factors, combined with the impact that mobile-related traffic may have on the fixed infrastructure, and the convergence of mobile and fixed services, drive towards a rationalization of the network architecture, resource allocation and both inter- and intra-operators' management procedures. Consequentially, from the perspective of service providers, the increase of mobile data traffic could require enormous new investments and efforts to expand and manage their networks, which are already optimized only for static wired data or voice services. The associated dimensions of space/time dependence of this traffic demand, "hostile" operation environment, unpredictable quality, and changing user's network attachment point represent major deviations with respect to the traditional communications paradigm. Unlike the classical pure wire-line network, the network supporting mobile data services should have different unique characteristics: enormous adaptivity in handing high-speed large traffic volumes, rapid changes of throughput variability, network congestions and re-routing, and the like. The mobile wireless service providers have started their own data services using their proprietary network and now they are trying to open their networks to the public internet domain. Thus, in the medium-term perspective, several wireless overlay networks will coexist, each with its own routing architecture, network management and support facilities, and their interworking in a unique IP-based backbone providing full support for all mobile service facilities will be a challenging objective. In this scenario, Mobile IP reveals to be the best mean to provide host mobility solution in these future converged networks and there are several proposals to incorporate IP-based technologies into the core networks of future wireless cellular systems such as UMTS [1] and Cellular IP [2]. Since the number of mobile users and terminals connected to these future systems would be very large, the scalability of the Mobile IP solution and of the whole backbone is of great concern and interest. Under the above circumstances, one of the more critical issues in mobile transport network architecture would be to build optimal traffic engineering policies to prepare for the heavy traffic growth and handle the great variability in the traffic flow distribution on the backbones generated from the increasing number of mobile users foreseen in the upcoming future. Based on this challenge, IETF (Internet Engineering Task Force) has standardized an evolutionary protocol, MPLS (Multiprotocol Label Switching), which gives us very flexible and intelligent opportunities to dynamically and efficiently route the data flows on traffic engineering and resource requirements basis, and to configure the protection paths for any resource failures. MPLS is actually playing a key role in delivering QoS and traffic engineering features in IP networks and can be considered a promising technology to enhance the ability of the network operators to control the network behavior while delivering mobile IP services, in accordance with customized service contracts (SLAs).

In this paper, we propose a framework integrating the IP Mobility and traffic engineering facilities in mobile data networks backbone on a common MPLS transport stratum. The integration improves the scalability of the Mobile IP handover and data forwarding process by leveraging on the features of MPLS, which are fast switching, dynamic traffic distribution, small state maintenance and high scalability. In addition, we have removed by means of the MPLS TE-tunnel/LSP facility the need for IP-in-IP tunneling from home to foreign network. Furthermore, our mobility framework has a very low header overhead, works with any IP version without the need of modifying the IP address when the network attachment point changes. This also avoids the well-known triangle routing problem. The proposed mechanisms have been experimentally investigated to evaluate how they can improve the performance and functionality of today's mobile data networks, and identify opportunities for further improvements. The paper is organized as follows: after a discussion on the quest for convergence in mobile data networks, a brief overview on MPLS and traffic engineering and their possible role is given. Then, we present our framework for the integration of Mobile IP and MPLS, along with the testbed layout and some experimental results. An outlook on further steps and open issues and complements the work.

2 Evolutionary Trends in Mobile Data Networking Architectures

Recent service demands in wireless communications has significantly changed, where the traditional focus was commonly limited to voice channels over wireless point-to-point connections between the base station and the wireless terminal/phone, thus not requiring any complex routing or switching networking topologies. With the introduction of mobile data services, the emphasis shifts from sheer coverage to the flexibility and functionality of the network. A large number of the current wireless service applications require broadband data communications as well as advanced wireless networking services. These advanced services require a new generation network architecture that is powerful and yet flexible enough to enable fast change.

2.1 State of the Art in Mobile Data Services

During the last years the Mobile Network operators have gradually begun to deliver data services to their customers. The first available solutions performing data transfer over the GSM network include the Short Message Service (SMS), which allows a basic e-mail exchange, and the traditional low speed Circuit Switched Data (CSD), which may be used to access Internet services. The main drawbacks of CSD are the very limited bandwidth capacity (9.6 or 14.4 kbps depending on the coding scheme) and the sub-optimal use of the radio interface. As a short-term approach to enable advanced data capabilities on current technology mobile handsets, GSM operators also deliver WWW-like services based on the WAP (Wireless Access Protocol) technology [5]. In order to provide more bandwidth, most GSM operators are deploying the General Packet Radio Service (GPRS) [3], which is a fully packet oriented technology designed to support both IP and X.25. A medium term evolution will be the migration to the 3rd generation mobile system (UMTS), which will be able to accommodate a wide range of data rates (from 144 kbps to 2 Mbps). The increasing diffusion of portable devices, such as laptops, PDAs and smart phones, has recently led to a further ongoing evolution that is the provision of wireless access to mobile full-featured Internet users, so that they can stay on-line even while moving, and take advantage of seamless user mobility. Available options include the use of cheap WLAN solutions (e.g. IEEE 802.11 [10], Bluetooth [11], HomeRF [12]) in indoor environments and the exploitation of the wireless coverage provided by existing satellite or cellular operators in urban and rural outdoor areas.

2.2 From the Overlay to the Converged Model

IP mobility services for wireless users are currently delivered over a variety of networks and technologies, including the above mentioned GSM, GPRS, satellite, wireless MAN, WLAN and many others. However, these solutions are targeted to a specific set of services and applications and have therefore different characteristics in terms of geographical coverage, bandwidth, delay, etc. As a consequence, transparent user roaming amongst different wireless networks will be the best way to deliver the widest range of services anywhere and in a cost effective way for both the user and the service provider. In particular, it is expected that the same geographical region will be covered by several wireless overlay networks, so that it will be up to the user to decide when to switch from one wireless access to the other based on availability or cost/performance considerations. That is worse, each of these overlay networks presents its own switching/routing architecture, QoS and SLA enforcement facilities, network management platform, and support staff. With the imminent evolution to the next generation wireless network, wireless service providers are striving to converge these separate infrastructures to a single network over a common packet core. Service providers that converge their fixed and mobile networks to deploy a more sustainable business model now will gain significant advantage over their competition, namely by:

- Reducing network complexity by putting all traffic onto a single architecture
- Simplifying the manageability (one network and service management platform).
- Cutting operational costs by provisioning and maintaining a single network.
- Concentrating efforts on a single layer, improving at the same time billing flexibility, customer profiling and fraud control.

Furthermore, the packet core that is implemented for 2.5G (GPRS)/3G (UMTS) and public WLAN services reflects the same type of architecture on which traditional wire-line data and voice services have been deployed. This creates further revenue generating opportunities that can enhance the wireless service offerings .

2.3 The Need for Traffic Engineering

In this kind of perspective, best effort/destination-based routing driven from classical IGP (Interior Gateway) routing protocols, exhibits serious disadvantages mainly because data packets typically travel only along the shortest available path, which might not be the most efficient resource usage in a continuously changing network state. Moreover, as to the traffic congestions from any unexpected failures of network elements, IGP protocols have been reported very poor performance in the aspects of fast re-routing or efficient load balancing. Overprovisioning is the simplest solution, but it will adversely affect the bottom line. On the contrary, the traffic engineering method requires more intelligent schemes for the network elements evenly distributing the traffic loads and pre-establishing backup paths. In detail, traffic engineering has as its ultimate objective the cost-effective dimensioning of network resources to handle the user's demand for telecommunications services - and hence the induced user information and signaling traffic streams. If the traffic engineering application implements the right set of features, it should provide precise control over the placement of traffic flows within the routing domain. Specifically, traffic engineering should provide the ability to move traffic flows away from the shortest path selected by the IGP and onto a potentially less congested physical path across the network. The ideal traffic engineering solution assigns network resources according to traffic requirements and only requires network capacity to be augmented in response to increasing traffic demands in a proportional and deterministic manner.

Although mobile services have been commercially available since the late seventies, traffic engineering for personal communications has been based - and to a great extent still is - on "current practice" of mobile operators, and has been dominated until recently more by radio transmission and coverage considerations rather than by classical traffic loading and service quality arguments. The justification for this has been that an elite customer base has traded-off impairments of service quality against ubiquitous service that is perceived as a major value [4].

3 Introducing MPLS in the Mobile Data Network Backbone

The mobile data network backbone must route dynamically and efficiently traffic over a technologically heterogeneous infrastructure. At the same time, it has to provide the same degree of reliability and control operators that are used to in their core networks. It should also endorse the use of traffic engineering techniques. According to these considerations, the ideal complement to an all-IP network is MPLS.

3.1 MPLS and Traffic Engineering Basics

MPLS is a packet forwarding technique being standardized by IETF [6]. MPLS uses labels to make forwarding decisions at the network node level, in contrast to the traditional destination-based hop-by-hop forwarding in IP networks. In MPLS, the space of all possible forwarding options in a network domain is partitioned into "forwarding equivalence classes" (FECs). For example, all the packets destined for a given egress may belong to the same FEC. The packets are labeled at the ingress depending on the FEC they belong to. Each of the intermediate nodes uses the label of incoming packet to determine its next hop, and also performs "label swapping," i.e., replaces the incoming label with the new outgoing label that identifies the respective FEC for the downstream node. The label swapping maps corresponding to the FEC are established using standard signaling protocols such as RSVP or CR-LDP. Such a label-based forwarding technique reduces the processing overhead required for routing at the intermediate nodes, thereby improving their packet forwarding performance.

Also, the label-merging procedure used by MPLS creates multipoint-to-point packet forwarding trees in contrast to a routing mesh in conventional network based on a similar paradigm such as ATM networks. This reduces considerably the size of forwarding table at the intermediate nodes, thereby improving their scalability. Another important capability that MPLS provides is that of "constraint-based routing." The ingress node can pre-establish a path/tunnel through the network that can be associated to a specific traffic flow. Once the path, called label switched path (LSP), has been created, traffic is mapped onto it according to the dynamic needs of the traffic and the capabilities of the path [7]. Every path may have dedicated resources associated with it and can be manually specified as explicit path or can be automatically created in a "best fit" manner, based upon the specified requirements. Explicit paths can potentially be chosen in response to many different criteria, including network traffic loads, available bandwidth, administrative costs, path delay, delay variation (jitter), hop count, cell loss ratio, etc. Rather than inefficiently carrying the explicit route in each packet traversing the path, MPLS allows explicit route to be carried only at the time LSP is set up using signaling protocols. The subsequent packets traversing this path are forwarded using packet labels. Constraint-based routing is potentially useful for traffic engineering [8]. The core component of the traffic engineering process is the ability to manage router resources, which is a requirement for every router, also called Label Switch Router (LSR), operating in the MPLS network. In addition, the resource capability and availability for each router in the network must be shared on a network-wide basis so that LSPs can be calculated via a centralized traffic engineering process, which resides in each MPLS-enabled edge router called a Label Edge Router (LER) .

3.2 A Perspective of Mobile Data Network Architecture

In order to propose possible architectural improvements, we first want to summarize the characteristics of a typical communication infrastructure deployed in mobile data-transport networks. The typical architecture of general Mobile Data Networks may consist of public and private subnetworks. The backbone may be built on high performance switches (typically ATM in legacy networks) or routers (core nodes) connected in a mesh (even full) of high speed data lines. Mobile nodes are attached to radio access subsystems or Base Stations (BS). A number of equal or even different radio access subsystems should be connected to a common Backbone through specialized concentrator nodes or Access Routers (AR). Whenever the coverage and capacity of a single radio access node does not suffice to fulfill the mobile connectivity requirements in a geographic area, we assume attachment of a sufficiently large number of access points to one or more ARs. Other specialized nodes, the gateways, connect the backbone to outside networks. In some simplified architectures AR, gateways and core backbone nodes can be collapsed in a single router/switch device. From the backbone perspective all gateways to access network subsystems can be modeled as legacy network nodes where traffic originates and terminates. The Backbone itself can be seen as one administrative domain entirely based on packet-switching technology. An access gateway may be connected to more than one core node. In that case the access gateway becomes ingress/egress point from the Backbone perspective. As in general packet-switching infrastructures, the network operation shall result in efficient resource utilization while providing the required degree of QoS and flexibility.

3.3 MPLS Benefits in Wireless Networks

By deploying MPLS in the core of the network, the entire transport infrastructure for mobile services can be straightforwardly converged onto a common multi-service packet core. The benefits of using MPLS for wireless network evolution include:

Evolution to an IP aware network. MPLS allows labels, and thus forwarding treatments, to be assigned to IP packets according to a variety of policies. In addition, the topology of the underlying transport network becomes visible to IP routing. The result is that different traffic types can be given the appropriate priority end-to-end across the packet network, something only previously possible if the core network was ATM or Frame Relay-based.

Protection of investment. By simply adding MPLS functionality at the control plane, the existing WAN switching (typically ATM) hardware can be re-used to transport IP traffic efficiently. These switches are usually configurable as ATM Label Switch Routers (ALSRs) and run an IP routing protocol as well as the MPLS label distribution protocol(s) to establish label switch paths over the existing infrastructure.

Gradual deployment. One of the most significant benefits provided by MPLS is increased scalability. Since an all-IP network able to support next-generation wireless services cannot be build overnight, operators may adopt a gradual deployment strategy and, at the same time, seamless network operation can be assured during the rollout.

Evolution to a GigaSpeed MPLS core. The next stage in the evolution of the network is to increase core capacity. One potential strategy is to migrate the current network infrastructure so that it is operating over a very high capacity MPLS-based network core. MPLS delivers a unified control mechanism for this evolution, as it has multi-protocol capabilities for running over mixed media infrastructures.

Improved Mobility support. Mobility can be addressed better if MPLS connections are available between the mobile access nodes: once a connection is established, roaming can be supported by rerouting the mobile connection upon hand-off. Using MPLS to set up the mobile connection and to provide for the required QoS tackles two key problems at once: mobility and QoS.

Support for CoS/QoS for service differentiation. MPLS defines the signaling mechanisms to support both Class of Service (CoS) and QoS. It provides the means to relate this to the DiffServ markings of the originating IP traffic. MPLS's ability to support constraint-based routing and traffic engineering delivers the QoS that is required to support Conversational, Streaming, and Interactive traffic, something only previously possible with ATM.

4 MPLS-Based Address Mobility

The proliferation of wireless LAN technologies and mobile terminals has prompted an increased need for efficient mobility management and seamless handover/roaming. The main drawback in the actual scenario is that mobility management is not performed at the IP layer but is handled almost completely by the underlying wireless infrastructure. This may lead to sub-optimal traffic routing and transport network utilization and does not allow seamless user mobility across different wireless media.

4.1 The Classic Mobile IP Addressing Model

The current Mobile IP approach [9] provides a standard method for node mobility within the Internet. Using the Mobile IP infrastructure, node may change their access point to the Internet without changing their IP-address. It is therefore possible to maintain transport and higher layer connections while moving. Mobility is just as suitable for mobility across heterogeneous media and the link by which a mobile node is directly attached to the Internet may often be a wireless link. A *Mobile Node* (MN) is identified by the IP address it has when it is in its home network, called its home address. When a MN moves away from its home network to a foreign network, it obtains a temporary *Care-Of-Address* (COA) from a specific registration *Foreign Agent* (FA) in the foreign network that is the MN current point of attachment. The MN registers with a *Home Agent* (HA), which is typically a router, in its home network, informing the latter of its COA. Any *Correspondent Node* (CN) wishing to communicate with the MN need not be aware that the MN has moved; it simply sends IP packets addressed to the MN's home address. These packets are routed via normal IP routing to the MN's home network, where they are intercepted by the HA. The latter encapsulates each such packet in another IP packet that contains the MN's COA as destination address. Thus these packets are delivered to the MN's new location by an IP-in-IP tunneling process. In the basic mobile IPv4 protocol, there is no direct routing from any correspondent node to any mobile node. Packets need to pass through the mobile node's home network and be forwarded by its HA, which is called the problem of "triangle routing." In IPv6 network, IPv6 nodes cache the binding of a mobile node's home address with its care-of address, and then send any packets destined for the mobile node directly to it at this care-of address [13]. All IPv6 nodes, either mobile or stationary, support communications with mobile nodes.

4.2 Performance Drawbacks in the Mobile IP Model

The current Mobile IP standard involves three different activities, which are the agent advertisement process, the registration process and the data forwarding process. It is crucial that these three different activities operate efficiently in order for the Mobile IP protocol to be scalable to systems consisting of huge numbers of mobile hosts. Instead, the operation of Mobile IP introduces a high control overhead and a lot of involved parties what can lead to delays of several seconds – inappropriate for voice or several other data streams. The main problem here is the number of signaling and authentication messages going back and forth over the potentially slow and unreliable wireless link, eventually throughout the world, from the MN to its HA. Also, the mobile data network may use several IP subnets for scalability reasons, so - at least for IPv4 - the MN must obtain a new link local address each time it crosses subnet boundaries. Getting a new IPv4 address may take several seconds; in IPv6 this may work slightly faster. Furthermore, all the packets directed to the MN are routed through the HA and for every IP packet that the HA receives, it needs to check if the destination IP address of the packet matches any MNs that are currently registered in a foreign network. If it matches, the HA will perform IP tunneling of the packet by adding an IP header to the packet and then sending it out to the routing process for forwarding. If no match is found, the HA just sends the packet out to the routing process for forwarding. The amount of processing required by the HA in this forwarding process depends on the number of MNs belonging to the home network that are currently registered in a foreign network. If there are many such MNs, the forwarding process will take very long. This poses a scalability concern that affects the use of the Mobile IP protocol in future wireless mobile systems and makes it definitively not well suited for fast or even seamless handovers.

4.3 A New MPLS-Based High Performance Mobility Model

There are numerous proposals that try to either optimize Mobile IP or use different ad-hoc mechanisms for a specific application domain. All of them require significant modification in standardized protocols of production technologies. Our mobility proposal is based on a fast and technology independent registration/handover schema using MPLS as a tunneling technology, and host routing entries that allows the mobile nodes to receive and send data at their new location without the need to get a new link local IP address (COA). The whole framework is based on the combined application of standard MPLS and IP Local Area Mobility without any modifications, such that it fits well into existing networks. In addition, the introduction of MPLS for the access router connections allows the use of its native QoS and traffic engineering features to dynamically adapt the connections to the load generated by mobile users while fulfilling the single mobile connections' QoS requirements. In our model, we assume a typical network providing multiple first-hop access routers which may have one or more wireless links or layer-2 networks with multiple wireless base stations (fig. 2) such that the AR is the first IP-capable network element seen from each MN. The network may also provide gateway nodes connecting the mobile data system with the outside world and one or more DHCP servers. Each AR acts as an MPLS edge router and Home and Foreign Agent. A full virtual mesh of MPLS LSPs is provided between all the ARs, so the overall number of LSP is $N(N\text{-}1)$ if there are N ARs. In addition, some other LSP will be required for connecting support nodes and gateways. The relative importance of those connections will be determined by the prevailing traffic pattern (MN-to-MN traffic might overbalance MN-to-gateway traffic). Nonetheless, traffic engineering techniques can be used to guarantee that the backbone stays well balanced. As many LSPs as needed may be set up between ARs serving highly populated areas. Backup LSP mapped to alternative routes may be reserved to handle link failures. The existing LSP re-routing mechanism guarantees dynamic adaptation of the paths to variations of the network state.

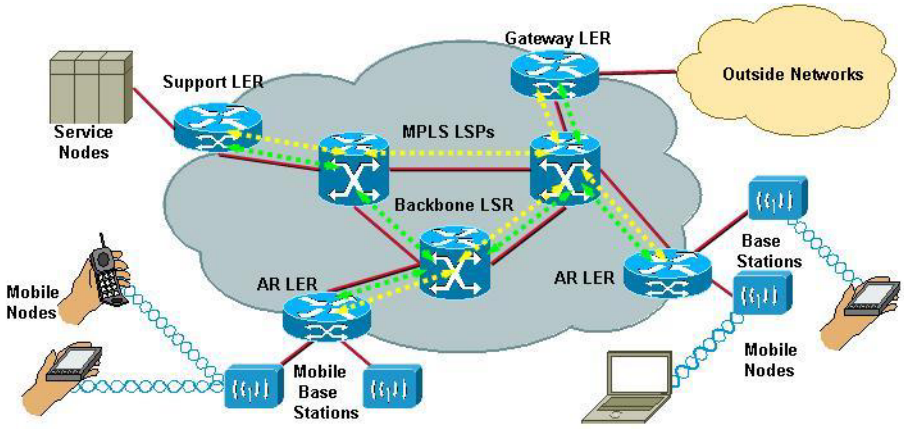


Fig 2. MPLS-based mobile data network architecture.

This approach greatly reduces header and signaling overhead in the whole mobile data network by avoiding the creation of multiple IP-in-IP tunnels, each associated to a mobile station and its Home Agent. Furthermore, the avoidance of the link local address change after each handover substantially simplifies all the handover/roaming procedure by dynamically routing the packets for the MN from the old Access Router to the new one with obvious performance improvements. From the addressing point of view maximum flexibility is achieved, that is a MN may have a statically or dynamically allocated IPv4 or IPv6 address, and addressing may be flat or using subnets.
In more detail, our mobility schema works as follows:

Initial Registration

Let's assume that a MN initially enters a cell attached to an Access Router that may be in the home network or in a foreign network from the addressing point of view. In both cases the MN will perform some sort of link-layer authentication (e.g. 802.1x/LEAP for WLAN) with the help of the BS and the AR; it receives a link local IP address associated to its home network and then it may perform a Mobile IP signaling (e.g. Mobile ARP) to establish the local host route. First, it should be noted that, unlike classic Mobile IPv4, the Access Router located in its home network will not assume a central role in all the future transaction but is considered like the other AR from the routing point of view. In the former case the MN can already send and receive IP traffic. If In the latter case the AR will work as a Foreign Agent and will trigger, if necessary, a label path establishment toward all the other AR and gateways on which it can do a binding update by announcing on its LSP mesh the mobile host route. Thus, the mobile host route is only known to the ARs and gateways, avoiding any unnecessary routing table growth in the other network nodes.

Handover

Once the association to the current BS is lost, e.g. the distance is too large for proper communication, the MN will scan the air interface for a new BS. If it finds one, it will register at layer 2 with that BS and then either wait for an ICMP Router Advertisement or it will issue an ICMP Router Solicitation. The according router is the AR where the BS is connected to. In any case, the MN examines the AR's IP address. We have to distinguish two cases here:

– The AR's IP address is the same as before (intra AR handover). This may happen if multiple BSs are connected to one AR. In this case, there's no need to perform any further steps, as there's no IP layer mobility at all.
– The AR's IP address differs from the old address, which means the MN has entered a new IP subnet. In this case, the MN must send a location update (Mobile IP signaling) message to the new AR. This message might be any IP packet because the new AR only needs to know which old AR the MN is coming from. This can be seen from the MN's source address subnet part. We propose to use a special Router Solicitation message or - if the link-layer supports it - a Mobile ARP request. The advantage is that the ARs need not scan the whole traffic to detect new IP addresses but receive location updates only when needed and only to a dedicated service. The new AR now looks if there's already a label path to the other ARs and gateways. If not, it will trigger a label path establishment from the ARs and gateways to itself (e.g. by using appropriate LDP messages). Once these steps have been completed, the new AR will also construct a host route entry for the MN and the according layer-2 interface. Then the AR has to send another location update message to the other ARs and gateways (mobile host route advertisement) telling them that the MN is now in its network. The ARs then will add a host routing entry for the MN pointing to the corresponding label path.

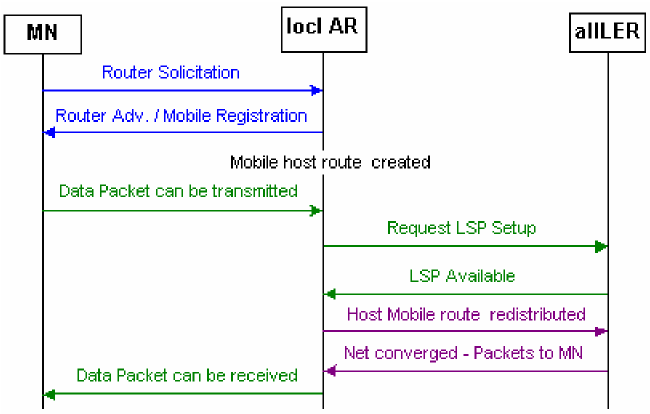

MN
locl AR
allLER
Router Solicitation
Router Adv. / Mobile Registration
Mobile host route created
Data Packet can be transmitted
Request LSP Setup
LSP Available
Host Mobile route redistributed
Net converged - Packets to MN
Data Packet can be received

Fig. 3. Handover scenario

At this point in time, the communication can proceed: packets from the MN will be sent over the new AR to the CNs using normal IP routing; the MN can reach the other MNs in its domain directly through the MPLS LSPs with their attachment's AR and the entire outer world through the LSPs connecting the mobile data network gateways. Packets towards the MN will run in the same way to the attachment AR via LSP from other ARs or gateways and finally reach the MN. The presence of MPLS LSP from an AR temporary hosting a MN and all other AR and gateways on the mobile data network is needed to avoid the triangle routing and provide the short-cut path.
The handover mechanism can be complemented, for improved security sake, by a fast Challenge-Response Algorithm for further authentication and authorization at the link layer.

5 Performance Evaluation

To evaluate the performance of the proposed model, we created a test network and designed a set of experiments to analyze the scheme. In detail, a simple MPLS testbed has been set up by properly extending the core of an existing experimental network made up by three Cisco Giga-Switch routers (12410 and 7606 models), assuming the role of backbone LSR, connected in a full mesh via Gigabit Ethernet interfaces, and two Cisco 7507 routers, operating as LER, that will be the mobile access concentrator routers. Two 802.11b wireless access points have been connected via Fast Ethernet to the ARs each through a catalyst 2924 L2 switch, as in Fig. 4 below. The access points have been properly located in our plant to obtain two semi-adjacent coverage cells with no overlap. A laptop equipped with an 802.11b WLAN interface has been located on the border of the first cell and quickly moved into the other cell to stimulate the handover process. Ethernet local area mobility has been used to realize the automatic setup of the host routes on the two cell AR.

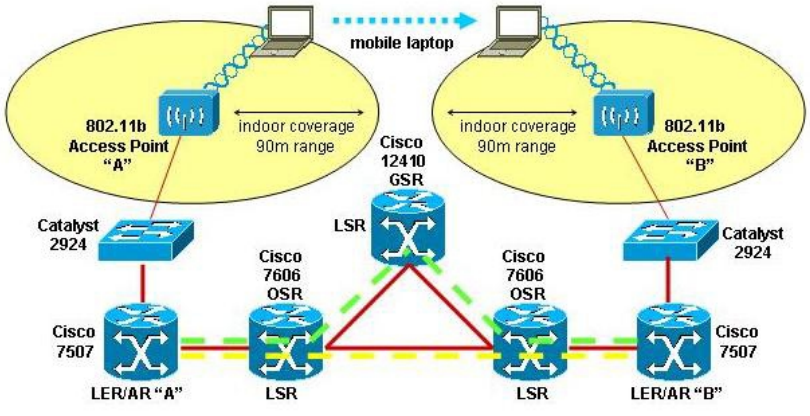
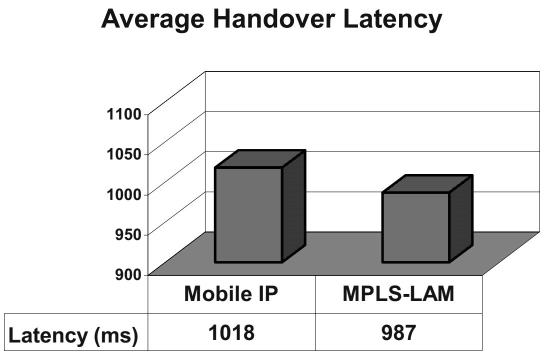


Fig 4. The testbed layout.

We performed both functional and performance comparative tests. The functional tests demonstrated the correct behavior of the whole framework during the handover when moving between the cells. We also extrapolated the handover time from the number of packet lost from a continuous ICMP ping from a fixed CN to the MN. Handover time has been measured also with a plain Mobile IP solution implemented on the same testbed and the results have been compared with the values obtained with our MPLS framework. The results, reported in the fig. 5 graph below, show a slight but significant decrement in handover time, that can be further improved by proper tuning actions. The reduction of Mobile IP handover latency is an extensively investigated subject (e.g. [14], [15]). However,. most of the proposed solutions involve rather complex extensions to the mobility support for IPv4 or IPv6.



	Mobile IP	MPLS-LAM
Latency (ms)	1018	987

Fig 5. Handover time - classic mobile IP vs. MPLS-Local Area Mobility.

6 Conclusions

By deploying MPLS in the core network, the entire transport infrastructure for mobile services can be converged onto the common packet core network. This enables immediate capital and operational savings. Furthermore, service providers that deploy converged data networks will gain significant advantage over other operators who decide to wait for an IP-only solution. These include time-to-market, cost savings and packet data experience. Service providers will also benefit from the stability of the existing ATM infrastructure, as an established, reliable packet core technology for the initial rollout of the network. The MPLS control plane will then allow evolving the packet core network to IP/POS without needing to immediately replace the ATM switches in which investment has already been made. In this context we introduced a new high performance IP-mobility framework integrating MPLS traffic engineering functionalities with local area mobility technologies. This integration makes IP-in-IP tunneling in the data forwarding process unnecessary; avoids the triangle routing and provide the short-cut path and QoS; it is built on top of well-known technologies that are currently available in a production environment. As COAs are no longer needed, IP address planning can be simplified. Since only minor changes to production hardware and software may be called for (mainly on the security side), rollout schedules and costs can be dramatically slashed, with competitive advantage. Finally, in our framework the mobility layer and the backbone layer are clearly separated. Technologies are applied at the level they were originally designed for, thus making network planning, rollout and operation more modular and manageable. Some issues call for further investigation. When handover occurs, packets in flight that had already been sent to the old AR are likely to be lost before redistribution. Some mechanism is needed to avoid route flapping in the spots where coverage areas of distinct ARs overlap. Finally, the growth of routing tables should be controlled.

References

1. P.C. Mason, J.M. Cullen, N.C. Lobley, "UMTS architectures", Mobile Communications Towards the Next Millennium and Beyond, IEE Colloquium 1996, Page(s): 401 -411.
2. Andrew T. Campbell, Javier Gomez, Andras G. Valko, "An Overview of Cellular IP", IEEE Wireless Communications and Networking Conference (WCNC'99), New Orleans, Sept. 1999.
3. "General Packet Radio Service (GPRS); Service description; Stage 1", ETSI GTS GSM 02.60 v6.1.0.
4. D. Grillo, "Personal Communications and Traffic Engineering in ITU-T -- The Developing E.750-Series of Recommendations," IEEE Personal Communications, vol.3, no. 6, December 1996, pp. 16-28. pp.647-656.
5. http://www.wapforum.org.
6. Rosen et al., "Multiprotocol Label Switching Architecture", IETF draft-ietf-mpls-arch-06.txt, August 1999.
7. R. Callon et al., "A Framework for Multiprotocol Label Switching", Internet Draft, draft-ietf-mpls-framework-02.txt, Sep. 1999.
8. Chuck Semeria, "Multiprotocol Label Switching: Enhancing Routing in the New Public Network," white paper, Juniper Networks Inc.
9. C. Perkins, "IPv4 Mobility Support", RFC 2002, Oct. 1996.
10. http://grouper.ieee.org/groups/802/11/index.html.
11. http://www.bluetooth.com.
12. http://www.homerf.org.
13. D.B. Johnson, C. Perkins, "Mobility Support in IPv6", IETF Internet Draft draft-ietf-mobileip-ipv6-14.txt, July 2000.
14. R. Koodli, "Fast Handovers for Mobile IPv6", Internet-Draft, September 2002.
15. K. El Malki et al., "Low Latency Handoff in Mobile IPv4", Internet-Draft, June 2002.

Dynamic Resource Management for Mobile Services*

Carmelo Ragusa, Antonio Liotta, and George Pavlou

Centre for Communication Systems Research, School of Electronics and Physical Sciences
University of Surrey, Guildford, Surrey, GU1 7XH, UK
{C.Ragusa, G.Pavlou, A.Liotta}@eim.surrey.ac.uk

Abstract. In this paper we propose a dynamic resource management system addressing the key requirements of mobile services – i.e. services realized as Mobile Agents (MAs). MAs are exploited here in different ways. First, to realize adaptable, configurable, context-aware services for 3G and beyond. Second, to develop a distributed monitoring system that suits the hurdles posed by service and network mobility. Finally, to construct a management system that can dynamically re-configure MA-based services for load-balancing and adaptation purposes. We present the resource management system architecture, a scheme providing run-time adaptation through agent mobility, and a prototype implementation along with some important simulation results.

1 Introduction

The increasing availability of relatively low-cost portable devices, accompanied by a rapid deployment of wireless and cellular infrastructures, has made mobile computing a reality. We are also witnessing how personal communications as well as business have become dramatically dependent upon networked computing systems relying on high-speed communication for multimedia and web-based services. The newly deployed 3rd Generation (3G) networks allows not only personal mobility and low-latency communication but also the so declaimed "always on" connections, which means access to advanced services in (almost) all situations while on the move.

In the meantime, the delay of the 3G launch has allowed sufficient time for the maturation of those sophisticated services that could not fully exist without 3G. Virtual Home Environment services, context-aware services and adaptable services are just examples of the enormous potential that can be unleashed in the 3G context.

Fundamental 3G services and applications have started to appear, but operators will only see a real return of their investments if they will be able to efficiently deploy more advanced services that exploit the full potential of 3G. Advanced services are also needed to justify further investments on communication infrastructures beyond 3G. Services and service platforms seem, therefore, the key element of the current

[*] The work reported in this paper has formed part of the WA1 area of the Core 2 Research Programme of the Virtual Centre of Excellence in Mobile & Personal Communications, Mobile VCE, www.mobilevce.com, whose funding support, including that of EPSRC, is gratefully acknowledged. More detailed technical reports on this research are available to Industrial Members of Mobile VCE.

telecommunication scenario and it is for this reason that so much attention is currently being dedicated to their effective realization.

Many problems still exist when it comes to deploying mobile services, that is services realized through software components or objects that are mobile and distributed. Mobile Agent (MA) technologies are good candidates for the development of such services. In the context of 3G, mobility assumes also a second dimension, that of node or terminal mobility. Software mobility can then be exploited in two different ways. The first one follows the conventional use of MA systems, which uses software mobility for applications such as user profiling, delegation of responsibility and so forth. A second important application consists of exploiting software mobility to overcome the issues arising from network and terminal mobility. The latter represents an attempt to deploy services in a way which guarantees an efficient use of the networked resources, despite of the dynamic changes in their status caused by node mobility.

Mobile services also have to overcome the hurdles arising from the peculiarity of mobile networks. They have to be resilient to Quality of Service (QoS) variations and temporary loss of connectivity. They also have to be designed in a way which accounts for the relatively higher cost of wireless links with respect to wired ones. Finally, they have to deal with the limited power and capability of mobile terminals. This implies that it should be possible to deploy resource-intensive services to users that are connected through thin clients.

The abovementioned requirements call for effective resource management solutions. We have been looking at the issues related to 'dynamic resource management for 3G services and beyond' within the Virtual Centre of Excellence in Mobile and Personal Communications (M-VCE) project [13]. In this paper we describe our approach and illustrate our initial findings and lessons learned. We have developed a middleware-based system, a logical service layer sitting on top of the network and providing an environment for the execution and management of MA-based mobile services.

While the M-VCE project has an ample scope ranging from low-level communication aspects to the development of 3G services, this paper is focused on those aspects related to managing MA-based services, mainly exploiting code mobility for dynamic resource balancing purposes.

In the rest of this paper we first give an overview our service execution environment, the Common Agent Location and Execution Environment (CALExE) that is further described in [3] [18]. We also outline the services and scenarios developed by the M-VCE project, as described in [14]. These are used here as reference cases for the dynamic resource management system (Section 3). Our results are presented in Section 4, followed by a survey of related work (Section 5).

2 CALExE & Mobile Services

The CALExE architecture and system ([3] [18]) is being developed as part of the Software Based Systems work area of the M-VCE project, a consortium comprising 7 UK universities and 28 companies representing the major international telco operators and service providers [13]. Software agents have been chosen as the key underlying technology for the development of services aimed at surviving the transition between

3G and 4G (the generation of networks that will follow 3G). Hence, great effort has been dedicated to the development of a suitable, that is efficient and secure, middleware layer for a controlled, managed execution of MAs.

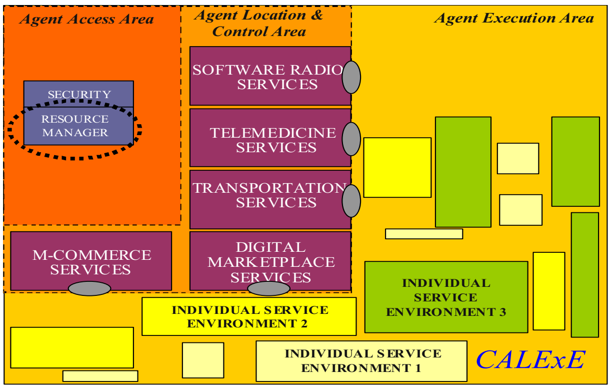

Agent Access Area
Agent Location & Control Area
Agent Execution Area
SECURITY
RESOURCE MANAGER
SOFTWARE RADIO SERVICES
TELEMEDICINE SERVICES
TRANSPORTATION SERVICES
M-COMMERCE SERVICES
DIGITAL MARKETPLACE SERVICES
INDIVIDUAL SERVICE ENVIRONMENT 3
INDIVIDUAL SERVICE ENVIRONMENT 2
INDIVIDUAL SERVICE ENVIRONMENT 1
CALExE

Fig. 1. CALExE system architecture.

The CALExE system architecture is depicted in Fig. 1. CALExE is a common platform where agents can be located and where they can interact to extend the variety of services available to mobile users. It provides computational resources as well as an environment for agents to execute and interact. CALExE offers management functionality for a controlled execution of agents in terms of performance and security. It is characterized by a modular architecture comprising three areas, each one related to particular aspects of the distributed system to be managed and providing important management functions for the various mobile distributed entities. These areas are:

- **Agent Access Area** is mainly devoted to controlling the access to resources and services provided by the platform. This area is responsible for the management of the physical resources under the control of the management platform. It is also responsible for the authorization and determination of access rights for the execution of different distributed entities, including both static and mobile agents. This area also manages the controlled access to remote resources and execution environments allowing, for instance, optimizing the execution of particular services by controlling the location of their sub-components or MAs. The resource manager enclosed in the dotted oval is described in further detail in the following sections.
- The **Agent Location and Control Area** is mainly concerned with the discovery, location, registration and control of the services provided and advertised by CALExE. This area is also responsible for monitoring the conditions in which the service is delivered – QoS monitoring is, for instance performed at this level. These functionalities are out of the scope of the paper (further details can be found in [3], [18])
- The **Agent Execution Area** provides a secure execution environment where agents can execute, negotiate, and act/react to particular events in the system. These

functionalities are out of the scope of the paper (further details can be found in [19] [20])

The dynamic resource management system described in Section 4 below uses as a reference the services and case studies that have been specified and are currently being developed by M-VCE. These are detailed in [14] [21] and briefly summarized below:

- **Home service** – Services in the home provide the user with a comfortable and secure environment, with easy access to features and intuitive responses, to enhance the user's productivity and general quality of life. Many tasks in and around the home can be performed by a set of co-operating mobile intelligent agents, acting on behalf of the user. An example scenario that highlights the emerging aspects of future generations wireless networks in the context of an automated home is where mobile terminals, wearable computers, appliances and personal area network (PAN) devices communicate with each other in an intelligent and autonomous way. Issues such as interoperability, uniform service provision, dynamic resource allocation and security pervade the whole of a distributed system.
- **Transportation** – The transportation scenario shows the chain of events during the travel of an executive. Travel consists of several stages where executive travels by different means of transport – car, plane, train and bus – continuously updating their travel details and ETA (estimated time of arrival) and using different services. Services can also be used autonomously by user representatives on behalf his/her behalf.
- **Mobile multimedia** – This scenario concerns the provision of multimedia, location-dependent or location-independent services. An example is and 'infotainment' services delivered to a user while on the move.
- **Telemedicine and Assistive Services** – Telemedicine provides an example of services, which have high importance but require ubiquitous availability. This scenario reflects the social role that mobile communications can play in achieving independent living for all. It has much in common with the first scenario but adds to it a human dimension.
- **Mobile commerce** Electronic commerce – fixed or mobile – between all combinations of business or people is hyped widely and loudly. Solutions exist for every scenario but it is not clear if they also solve known problems, or if they raise new issues yet to be addressed.

Fig. 2 shows an example of a scenario where a user requires a mobile service through a User Agent (UA). A mobile service can be composed of 'static' and 'mobile' components. While the static part of the service can be configured and renegotiated in terms of computational resources the mobile one can also be moved to exploit better resource availability. In the picture a number of CALExE subsystems, (or simply CALExEs) are distributed across the networked system while also being monitored by the monitoring agents. CALExEs can either be positioned in routers or, more realistically, in their vicinity in order to exploit the knowledge that routers have of the network. Each monitoring agent will monitor one or more CALExEs within its area of competence, holding information on their position and resource availability. Monitoring agents also know the location of the other peer-entities, forming an overlay logical monitoring network. Fig. 2 shows an example where the UA, after reaching the nearest CALExE, interacts with the static agent. From there, the MA can subsequently decide to autonomously migrate or, alternatively, it can be forced to migrate by the resource manager, as illustrated in Section 3.

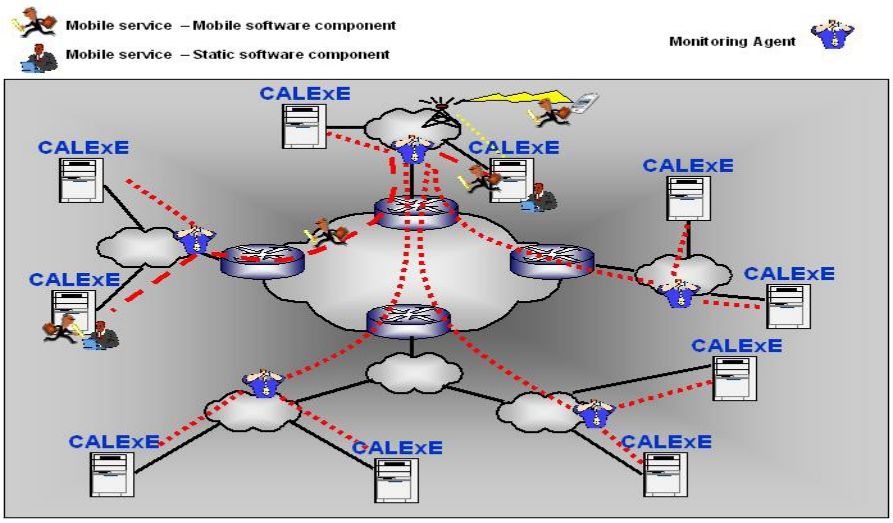

Mobile service – Mobile software component
Mobile service – Static software component
Monitoring Agent
CALExE

Fig. 2. Mobile Service example.

It may be important mentioning that the above scenarios are not meant to cover all possible situations; they have been specified with the aim of illustrating example applications and use them to extract requirements for the design of the CALExE middleware system. Each scenario has been used to identify specific issues, constraints and requirements on CALExE and, in particular, on the resource management system described below.

3 Dynamic Resource Management System

3.1 System Requirements and Functionality

The CALExE system consists of a number of distributed CALExE sub-systems that replicate some or all of the functionality of Fig. 1. For instance, the execution environment and the functionality exposing local resources to the resource manager should be present at every node running agents/services. The resource manager will use relevant information about the availability of distributed resources to trigger a variety of load-balancing actions. A typical action is, for instance, the reconfiguration of a service through the dynamic re-location of one or more MAs, in response to a scarcity of resource in current MA locations. Another example is the location of MA service components in a CALExE sub-system that acts as a proxy to a thin mobile terminal that could not otherwise execute a resource-intensive MA.

The resource manager plays therefore the key task of configuring the mobile services, matching the service resource requirements with the run-time resource availability. It also provides critical run-time resource availability information to those MAs that enjoy a certain level of autonomy and can, hence, self-relocate for load-balancing purposes. Therefore, the manager will not only manage the access and allocation of computational resources within each individual execution environment, but also determine, in real-time, the particular location where each service (or MA component) should be executed. In this case, the management ability is decoupled between the agents, that carry their service tasks, and the Dynamic Resource Management system that triggers agent migration for load balancing purposes.

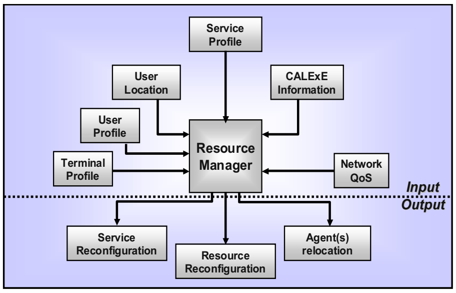


Fig. 3. System Input/Output model.

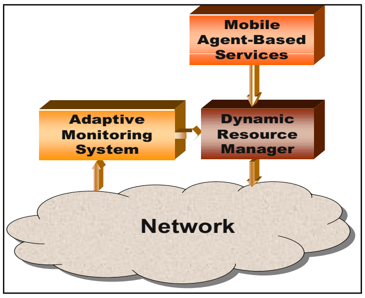


Fig. 4. Management System.

CALExE supports context-aware, adaptable services and has, thus, been designed to consider contextual information as a way to trigger service adaptation, again, through agent mobility. The input parameters to the resource manager are summarised below (Fig. 3):

- **User profile & SLA**: The user preferences and quality expectations are mapped onto an SLA (Service Level Agreement) agreed among user, service provider and network operator (NO).
- **User Location**: if the service is offered by a different entity that the NO, the NO should provide user location information which is fundamental to most context-aware, location-driven services. The recent developments in the process of standardizing open network interfaces (e.g. OSA/Parlay, web services) suggest that this requirement is feasible [22].
- **Terminal profile**: the knowledge of terminal capability is fundamental to most adaptable services in order to establish the best way of delivering the service. Memory or other terminal resource constraints are used as an input by our resource manager in order to best configure the service. Terminal capability information can be dynamically obtained using, for instance, the CC/PP protocol [23].
- **Service profile**: this input provides a broad model of the service requirements in terms of resource consumption (e.g. computation, storage, memory, QoS, etc). This information is associated to services on the basis of an accurate statistical analysis.
- **CALExE Information:** each CALExE subsystem includes an execution environment and an associated state/availability of the resources. Information collected from each CALExE sub-systems:

  - **CALExE Location**: the management system uses the location of execution points when deciding on how to locate/re-locate MAs for load balancing purposes. Latency minimization is achieved by choosing appropriate nodes for MAs.
  - **CALExE Workload**: the workload of each execution environment has to be supplied to the management system in order to extract the computational resources available and trigger due actions.
  - **Local routing table**: routing tables are used to indirectly get the status of the network and estimate distances/latencies among CALExE subsystems.
- **Network QoS**: this relies on network layer mechanisms such as DiffServ or MPLS aimed at supporting the required QoS.

The effects of the manager can manifest themselves in different forms, depending on the value of the above set of input parameters (Fig. 3). One possibility is to re-configure the service, for instance, by renegotiating its QoS requirements in face of QoS degradation. QoS can be pursued through management policies aimed at preserving the overall system and network resources, while meeting the range of QoS levels acceptable to the user.

For this reason, in CALExE services/MAs are allocated a specified amount of resources, in terms of computation, memory, storage and network resources. Another possibility is to react to limited QoS by requesting an increased level of resource allocation to the CALExE system.

An alternative to service and resource reconfiguration is given by run-time MA relocation, that is triggering the migration of one or more MA service components. This operation is aimed at load-balancing the resources used across the CALExE subsystems.

Our approach to dynamic resource allocation and agent location is described in the following two sections. The overall system functionality, as depicted in Fig. 4, consists of two main blocks, the monitoring system and the resource manager. Our services rely upon the latter that, in turn, relies on an efficient monitoring system capable of coping with the dynamics and scale of the CALExE environment and its MA services.

3.2 Adaptive Monitoring System

The requirements that the CALExE middleware poses on the monitoring system are quite stringent. First, the resource management system relies on timely information including the state of the resources available through CALExE. Service configuration and MA location are in fact closely related to the availability of network, storage, memory and processing resources across CALExE subsystems. Each of those resources dynamically varies over the time, depending on the service load, user location, user QoS demand and so forth.

Our monitoring system keeps track of all the above parameters, feeding their values to the management system both periodically and following and event-based approach. Scalability and dynamics are therefore the challenges faced by the monitoring system.

After considering various approaches, we finally decided to develop our own monitoring system that could address those challenges. Because we could rely upon an MA execution environment (i.e. the one used for MA services) we were able to design a fully MA-based monitoring system that would combine the advantages of distribution, activeness, adaptability, and reactivity. We have already reported our MA-based monitoring system in [2], assessing its scalability and adaptability to changes in the network state (e.g. faults, congestion, etc). Therefore we shall only give here a brief overview of the system (the interested reader can find more details in [2]).

In our system, monitoring of fixed or mobile resources (including nodes, MA service components, CALExE subsystems etc) is carried out by MAs. The monitoring process consists of two phases. Initially we need to deploy the monitoring MAs, depending on the status of the network and on the location of the target objects to be monitored. This is equivalent to partitioning the monitored system into sub-partitions that are, in turn, monitored by individual MAs. Our approach to the calculation of the number of agent and their location is to exploit information that is readily available in the system. Our monitoring system relies on the routing tables stored in the network, which are maintained up-to-date by the network routing protocols

Subsequently, upon agent deployment the agent system needs to be able to self-regulate in order to adapt to changing conditions. This is achieved by triggering agent migration in a controlled fashion to avoid instability due to continuous agent migration. Again, agents compute their location on the basis of routing information that, in turn, reflects the network state.

3.3 The Resource Manager and Run Time Service Adaptation

The resource manager is a core component of the CALExE middleware and is present in each CALExE subsystem. It performs the important tasks of *admission control*, *resource allocation*, and *resource discovery*. In order to describe its operation, let us suppose that we want to deploy a mobile service, composed of a combination of static and mobile agents. Fig. 5, illustrates the various steps starting from the submission of a job request (i.e., the intention to deploy a service) until service deployment. It depicts the logic executed at two different CALExE subsystems. The 'local' CALExE is the one that receives the job request in the first place. The 'remote' CALExE represents a generic candidate subsystem for the execution of some components of the mobile service under scrutiny.

The execution of service components within CALExE subsystems has to be endorsed by their respective resource managers. When a job request is received by a CALExE subsystem, this goes through an *admission control* procedure aimed at establishing whether sufficient resources are locally available. Admission control relies on information provided by the monitoring system and attempts a matching between the service profile (in terms of estimated resource requirements) and the local resource availability. It should be mentioned that service requirements might also specify that a maximum latency or a minimum throughput must be guaranteed between two different service components.

Those service components that pass the admission control procedure are instantiated and executed locally, following an appropriate *resource allocation* procedure. Static agents will, then, be forever bound to the particular CALExE subsystem that has initially accepted them. Instead, MAs can still be re-located at run time, either as a result of an action triggered by the local manager (e.g. because of scarcity of local resources) or as a result of a decision autonomously taken by the MAs themselves.

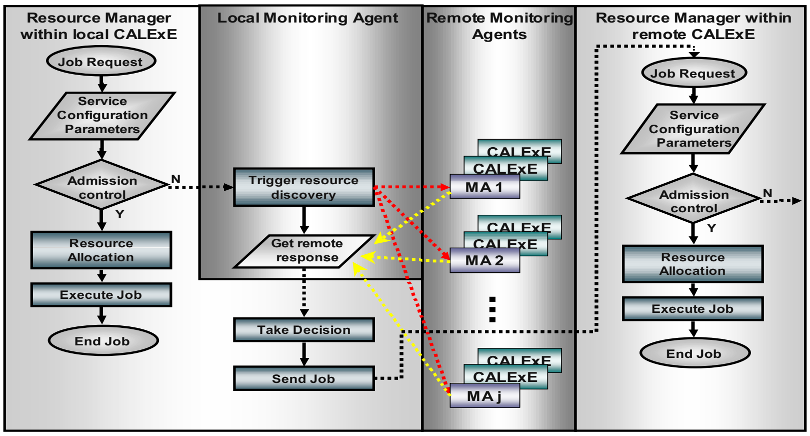


Fig. 5. Resource management system.

Any job that is not admitted locally triggers a *resource discovery* procedure through which the local CALExE finds out about alternative subsystems that have the potential to admit those jobs. Again, the monitoring system realized through local and remote monitoring agents (MA_1, MA_2, MA_j,) assists and provides information during this phase.

The discovery procedure will generally result in one or more candidate CALExE subsystems for each of the candidate jobs (i.e. MAs in search of a CALExE). The local manager can then employ different strategies aimed at selecting the best match between jobs and CALExE subsystems. We have considered three of them:

1. **Fastest first**: in this case each job is matched with the CALExE subsystem that responds first. The fastest reply may be received by the entity that is logically closer to the requestor. This may therefore be useful when trying to minimize latency between local and remote CALExEs.
2. **Largest availability**: in this case jobs are matched with CALExEs holding the largest availability of resources. This may assure an even utilization of resources across the system.
3. **Compromise distance-availability**: this solution tries to compromise between the above approaches, seeking a good load balance as well as latency minimization.

In such a way the local manager eventually delegates each of the pending jobs to the appropriate remote CALExE subsystems that will, again, put the job through the same admission/allocation/discovery procedure (right hand side of Fig. 5). MA delegation will continue until a suitable, unloaded CALExE subsystem is found. In our simulations this process tends to terminate after one or at most 2 iterations, depending on the level of load of the overall system. Clearly a heavily loaded system will require a longer MA allocation procedure.

4 System Evaluation

4.1 Methodology

The evaluation of the above resource management system has been carried out through simulation, developing three essential ingredients: a model of mobile services; an implementation of the algorithm of Fig. 5; and a design of realistic network conditions.

Our service model reflects the creation of composite services with static and mobile components/agents, the interactions among agents, and the resource utilization by the agents in terms of network links, CPU cycles, memory, and disk. Our service model reflects the requirements and constraints of the five services that are currently being developed within the M-VCE project and are documented in [14]. In this way we shall eventually be able to assess our resource management system for those services.

In order to carry out simulations under realistic network conditions, the algorithm has been implemented over the JavaSim network simulation which provides the underlying support for wired and wireless networking [15]. IP and TCP have been adopted as the basis for simulating routing and transport protocols, respectively. We have also enhanced the simulator with MA capability needed for the realization of the monitoring system and for the deployment of MA services. Functionalities include agent creation, cloning, migration, termination and so on.

The GT-ITM topology generator [16] was used to create realistic, Internet-like, transit-stub topologies. The simulations have been executed for topologies of 25, 40, 75, 100, 150, 200 and 300 nodes. To ensure statistical significance, simulations have been repeated 20 times for each of the above network sizes. That is, we have randomized the simulation process, generating families of 20 topologies at a time, characterized by comparable topological features (e.g. average node degree, number of nodes, etc). In this way we could assess the sensitivity of our metrics to network size, using the number of node as network size parameter and isolating other effects.

An important parameter of the system is its ability to adapt to variations in the system load. The management system achieves that by dynamically monitoring system resources and triggering service reconfiguration or MA migration. The metric used to assess the system performance was the total hop distance, computed as the sum of all the hops or 'link legs' involved in the service communication among all the service components. For instance, if the service is realized through two intercommunicating MAs, the total hop distance would be the number of links that interconnect those MAs. As the system/network evolves, the distance between those agents may increase, e.g. as a result of link failure, congestion and re-routing. It is the management system responsibility to trigger appropriate re-location, leading to reducing overall distances and, consequently, reducing overheads.

4.2 Results

Fig. 6 illustrates the behavior of a typical mobile service for an increasing number of congested links. We have measured the overall total hop distance (i.e. involving all service components) in two different cases: a) service realized with static agents only (no run time adaptation through agent migration); and b) MA based services (dynamic MA relocation triggered by the management system).

As expected, system congestion results in an increase in total hop distance which, in turn, means an increase in traffic and response time. Our adaptation strategy, however, succeeded in significantly reducing the rate of performance degradation (as indicated by the difference in slope between the two best linear fit lines).

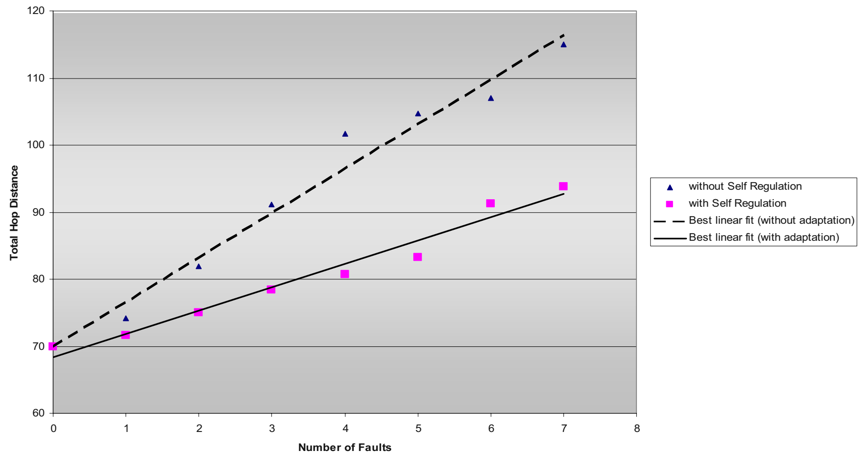


Fig. 6. Performance results of the resource management algorithm.

Fig. 6 depicts the results for a 40-node system and a family of Internet-like topologies. However, when we repeated the experiments for larger systems and for a variety of other scenarios the results were qualitatively equivalent.

The background load of the system and its overall congestion significantly affected the service initial deployment time only under extreme conditions, i.e. when the amount of resources available through CALExE could not meet the service resource demand. Instead, under average loading, MAs were mostly deployed in 1 or 2 iterations, depending on the relative number of CALExE subsystems with respect to the size of the system.

5 Related Work

Various researchers have used MAs as well as distributed resource allocation techniques to support services. In [1] the use of MA technology in a wireless environment to address QoS issues through a load balancing solution is described. The approach performs load balancing of processes in different servers, assures a QoS to specific classes of users and supports an algorithm to predict the user movement within different cells. Advantages of this work include the agents' capability to trigger adaptation of applications on behalf of the customers and the implementation of resource-state-based policies in a dynamical fashion. It is worth mentioning that this solution works more at network level, while the work presented in this paper is at service level.

In [4], the authors present a policy-based software architecture that implements application QoS management. The approach is dynamic in that the application is started with an initial resource allocation that can be dynamically incremented by the manager if needed. One limit highlighted by the authors is that if there are several applications on the same host with QoS requirements, the corrective actions may not be feasible.

In [5], the authors provide a resource allocation technique using a multi-agent system called *Challenge*". Challenger consists of agents that individually manage local resources. These agents communicate with one another to share their resources for an efficient use of the system. Although generally useful for resource allocation in distributed systems, the focus of the work is not aimed at systems with tight real-time requirements.

Smart Reminder shows how the interoperation of different agents focused on single tasks can realize complex applications [6]. This system is composed of a set of personal agents that support their user in all situations where people come together spontaneously, e.g. an encounter on a corridor. So the system tries to establish a framework where different autonomous agents can easily work together to deliver aggregated benefits to their users. The authors claim that these systems must be as autonomous as possible so that the time and effort saved is not countered by excessive co-ordination overheads.

The Odyssey architecture consists of a set of extensions to operating systems to support mobile, adaptive information access applications [7] [8]. Its central idea is that the system monitors resource availability, notifies applications of the relevant changes and enforces resource control. This approach extends the system-level functionality working therefore mostly at operating system level.

Paper [9] presents a middleware architecture called Agilos, which is used for QoS adaptation, achieved at two levels through the components adaptor and configurator. The adaptor works at system level and is in charge of specific type of resources, e.g. CPU or network bandwidth. In this way a fair and stable distribution of the available resources among concurrent applications is ensured. The configurator instead works at application-level and serves one application at time. After getting applications information it maps adaptation decisions made by the adaptor to application-specific parameter-tuning or reconfiguration choices within the application.

The TLAM (Two Level Actor Machine) is a two level model, which uses base actors and meta actors [10]. Base actors carry out application level tasks while meta actors are part of the runtime system, which provides a set of core services for distributed systems. These are, for example, communication, resource management, remote creation, migration, load balancing, scheduling, synchronization, and replication. Meta actors can access and change information within a base actor.

RAJA is an agent infrastructure for resource-adaptive systems that addresses resource-adaptivity by promoting agent interoperability [11]. Adaptation decisions can be negotiated between the agents or made corporately at runtime. It is a multi-level reflective architecture, which separates non-functional resource management from application functionality. The distinction between basis agents and their controllers allows structured programming and eases the transformation of resource-unaware legacy systems into resource-adaptive systems. The separation between controllers and meta agents isolates application- from system-level resource management and advocates autonomy of agents.

Finally, Grid gives the possibility of using networks of computers as a single, unified computing resource [12]. It is possible to cluster or couple a wide variety of resources including supercomputers, storage systems, data sources, and special classes of devices distributed geographically and use them as a single unified resource, thus forming what is known as a "computational grid". Within Grid resource managers or applications must tailor their behavior dynamically so as to extract the maximum performance from the available resources and services. The grid resource management systems must dynamically trade for the best resources based on a metric of the price and performance available and schedule computations on these resources such that they meet user requirements.

The abovementioned work attempts to address the requirements of future 3G services. Some work does not use the mobile agent technology, which makes it different from our approach. When the agent paradigm is used their approach relies either on agent autonomy or on multi-agency. The CALExE middleware can be seen as combining autonomy with more conventional distributed management.

6 Conclusion and Future Work

The latest technological advances in the area of fixed, mobile and cellular communication are opening the avenue of advanced services that were unimaginable just a few years ago. In this paper we have looked at some of the issues related to effective service control. We have taken the perspective of the MA technology as a key enabler of mobile service realization and control, focusing on important agent location control aspects.

Our work started from the assumption that in an increasingly mobile environment, services are mobile and so should be the control and management system. We started by developing an MA-based distributed, adaptable monitoring system, as reported in [2]. We then moved onto designing and prototyping some reference advanced services in the context of the M-VCE project [13], leading to a more abstract model that would be suitable for evaluation purposes. Mobile services are naturally implemented as a combination of static and mobile components. Hence, MAs are one of the best candidates for service realization. Having come to this conclusion, we are then open to interesting ways of controlling and configuring services in such a way as to make use of distributed resources that should always be conserved for scalability purposes.

This paper represents an important first step of our investigation of a hybrid, service management approach. We are looking at how to combine agent mobility and autonomy with a more conventional approach in which it is the management system that triggers management actions. Mobility, autonomy and system-initiated management are not always compatible and may lead to instability or integrity problems. Some of these problems have already arisen during our experimentation work and will surely require further attention. However, our initial results of the proposed resource management algorithm indicate that MA-based service management is certainly an interesting research avenue.

References

1. G Anastasi, et al., *An agent-based approach for QoS provisioning to mobile users in the Internet*, Proc of SCI2000, Orlando (Florida-USA), July 2000.
2. A. Liotta, G. Pavlou, G. Knight, *Exploiting Agent Mobility for Large Scale Network Monitoring*, IEEE Network, special issue on Applicability of Mobile Agents to Telecommunications, Vol. 16, No. 3, IEEE, May/June 2002.
3. O Lazaro et al., *Management System Analysis – Security, Performance and Integrity Issues*. Deliverable D1.4 M-VCE project, (www.mobilevce.com), Oct 2002.
4. G Molenkamp, et al., *Distributed Resource Management to Support Distributed Application-Specific Quality of Service*, Proc of the 4th IFIP/IEEE International Conference on Management of Multimedia Networks and Services, Chicago, Illinois, October 2001.
5. A. Chavez, et al., *Challenger: a Multi-Agent System for Distributed Resource Allocation*, Proc of the 1st International Conference on Autonomous Agents, p. 323-331, 1997.
6. F. Kargl et al., *Smart Reminder - Personal Assistance in a Mobile Computing Environment*, Workshop on "Ad hoc Communications and Collaboration in Ubiquitous Computing Environments" at CSCW 2002, New Orleans, USA, Nov. 2002.
7. B. Noble, *System Support for Mobile, Adaptive Applications*, IEEE Personal Communications, February 2000.
8. B. Noble et al., *Agile Application-Aware Adaptation for Mobility*, In Proc. 16th ACM Symposium on Operating System Principles, 1997.
9. B. Li et al., *Middleware QoS Profiling Services for Configuring Adaptive Applications*, In Proc. of Middleware 2000.
10. N. Venkatasubramanian, C. Talcott, *Reasoning about Meta Level Activities in Open Distributed Systems*, In Proc. of ACM Principles of Distributed Computing, 1995.
11. Y. Ding et al., *a resource-adaptive java agent infrastructure*, Proc of Agents 2001, Montreal, Canada, 2001, pp. 332 – 339.
12. M Baker et al., *The Grid: International Efforts in Global Computing*, International Conference on Advances in Infrastructure for E-Business, Science, and Education on the Internet, SSGRR2000, L'Aquila, Italy, July 31-August 6, 2000.
13. Mobile Virtual Centre of Excellence (MVCE) project. www.mobilevce.com
14. Turner, P. and Lloyd, S. (Eds.) *Agent System Specification*. Deliverable R5, M-VCE project, (www.mobilevce.com), May 2001.
15. The JavaSim simulator, www.javasim.org
16. Source code of GT-ITM, available at http://www.cc.gatech.edu/projects/gtitm/
17. W. Tuttlebee, *Mobile VCE: the convergence of industry and academia*, IEE ECEJ, 12 (6), 245-8, December 2000.
18. O. Lazaro, J. Irvine, D. Girma, J. Dunlop, A. Liotta, N. Borselius, C. Mitchell, *Management System Requirements for Wireless Systems Beyond 3G*, Proc. of the IST Mobile & Wireless Telecommunications Summit, Thessaloniki, Greece, 16-19 June 2002.
19. Borselius, Security for agent systems and mobile agents, in: C. J. Mitchell, editor, Security for Mobility, IEE Press (to appear).
20. N. Borselius, Mobile agent security, Electronics & Communication Engineering Journal, October 2002, Volume 14, no 5, IEE, London, UK, pp 211-218.
21. Turner, P. J. *et al*, (2002) Scenarios for Future Communications Environments. Technical Report ECSTR-IAM02-005, Department of Electronics and Computer Science, Southampton University.
22. The Parlay Group, Parlay Web Services, Overview Status, Public Version 1.0, 31 October 2002, www.parlay.org
23. W.Okada, et al, *Applying CC/PP to User's Environmental information for Web Service Customization*, Tenth International World Wide Web Conference, June 11, 2001, Hong Kong.

Adaptable Mobile Applications: Exploiting Logical Mobility in Mobile Computing

Stefanos Zachariadis, Cecilia Mascolo and Wolfgang Emmerich

Dept. of Computer Science, University College London
Gower Street, London WC1E 6BT, UK
{s.zachariadis,c.mascolo,w.emmerich}@cs.ucl.ac.uk

Abstract. An increasing number of applications is being written for mobile hosts, such as laptop computers, mobile phones, PDAs etc. These applications are usually monolithic, featuring very limited interoperability and context-awareness and are usually difficult to deploy and update. Application engineers have to deal with a very dynamic set of environments that these applications are in contact with and it is becoming increasingly difficult to design an application that will be able to cater to all the user's needs in those environments. This new setting forces a shift from design-time to run-time effort in developing software systems. To solve these problems and to allow a new class of ubiquitous and adaptable applications to be built, we have designed and implemented SATIN, a middleware system that allows the flexible use of logical mobility techniques by applications running on mobile hosts which are connected to very different networks. In this paper we describe our approach and show how SATIN can be used to deploy and update applications on mobile devices easily and efficiently.

1 Introduction

With the recent developments in wireless networks (Wavelan, Bluetooth) and the sales of mobile computers of any kind (such as laptop computers, Personal Digital Assistants (PDAs), mobile phones etc.) soaring, we are experiencing the availability of increasingly powerful mobile computing environments which are exposed to an increasingly dynamic setting. As such, a new highly mobile scenario for mobile devices and applications is being created, potentially allowing for interaction with and adaptability to any changes in their setting, or the development of *context-aware* applications. The major characteristic of this scenario, is heterogeneity, in the software, hardware and networking levels as the devices that form it are composed of a large number of different applications, middleware systems and hardware and can access different networking infrastructures. This new setting forces a shift from design-time to run-time effort in developing software systems. The current industry state of the art proposes monolithic applications which feature little to no interoperability, forcing application developers to anticipate at the design stage what possible uses their software will have throughout its lifetime. To tackle the issues arising from such heterogeneity, there is a need to create software systems that can automatically adapt to tackle changes to the environment and to users' needs. We have also recently witnessed the acceptance of logical mobility (LM) techniques, or the ability to ship part of an application or even a complete process from one host to another. As such, LM techniques have been successfully used to enhance a user's experience (Java Applets), to dynamically update an application (Anti-Virus software etc.), to utilise remote objects (RMI, Corba, etc), to distribute expensive computations (Distributed.net) etc. Whereas various mobile middleware systems have been developed, the use of LM in those systems has been very limited. We wish to show that providing the flexible use of LM primitives to mobile computing applications through a middleware system, will allow for a greater degree of application dynamicity, will provide new ways to tackle interoperability and heterogeneity as well as ease deployment. Section 2 continues with an introduction to LM as well as a summary of limitations of related work. Section 3 presents a case study for deploying applications on cellular phones. Section 4 identifies and describes the principles of our approach, while Section 5 describes how it can be used.

2 Background and Related Work

LM (LM) is defined as moving parts of an application or migrating a complete process from one processing environment to another. It has been classified[5] into the following set of paradigms: *Client - Server* (CS), a popular paradigm in traditional distributed systems, dictates the execution of a unit of code in a server, triggered by a client, which may receive any result of that execution. The most common example of this paradigm is the use of Remote Procedure Calls (RPCs). *Remote Evaluation* (REV) dictates that a host sends a particular unit of execution to be executed in another host. A result may or may not be needed, depending on the application. This paradigm is employed by Distributed.NET, Seti@Home and other similar distributed computing environments. In the *Code on Demand* (COD) paradigm, a host requests a particular unit of code from another machine, which is then shipped to the original host and executed. This is an example of dynamic code update, whereby a host or application can update its libraries and available codebase at runtime. Many examples of COD have recently emerged, due to the popularity of Java and its built-in class loading mechanism and object serialisation framework. A *Mobile Agent* (MA), is an autonomous execution unit. It is injected into the network by a host, to perform some tasks on behalf of the user or an application. The agent can migrate from a processing environment or host to another. The application of LM in Mobile Computing context has, up to now, been quite limited. Some research investigating the subject has, however, been carried out. Current efforts into this area can be roughly grouped into two categories: Approaches which use LM to provide reconfigurability in the mobile computing middleware itself, allowing applications to interact with services provided by heterogeneous platforms and middleware systems, and approaches that use certain paradigms of LM to provide particular functionality to applications. Examples of the first category include ReMMoC[3], a middleware platform which allows reconfiguration through reflection and component technologies. It provides a mobile computing middleware system which can be dynamically reconfigured to allow the mobile device to interoperate with any middleware system that can be implemented using OpenCOM components. UIC[8], another example in the first category, is a generic request broker, defining a skeleton of abstract components which have to be specialised to the particular properties of each middleware platform the device wishes to interact with. Examples of the second category include Lime[7], PeerWare[4] and Jini[2]. Lime is a mobile computing middleware system that allows mobile agents to roam to various hosts sharing tuple spaces. PeerWare allows mobile hosts to share data, using REV to ship computations to remote sites hosting the data. Jini is a distributed networking system, which allows devices to enter a federation and offer services to other devices, or use COD to utilise services that are already offered. The problem of these approaches with respect to our work is that their use of LM is limited to solving specific problems of a limited scope. For example, Jini uses COD to offer services, and PeerWare uses REV to distribute computations. What the middleware system described in this paper does, is to provide the flexibility to offer the solutions that previous approaches do, but its use is not limited to these, as will be made clear in the following sections.

3 Case Study: Deploying Software on Mobile Phones

Mobile phones are becoming increasingly powerful: State of the art phones feature a fast CPU, large amounts of memory, are usually equipped with a number of networking interfaces, like IrDA, Bluetooth and GSM/GPRS and can mediate packets between the different networks. Hence, in addition to being able to connect to the network operator, modern phones have the ability to form Ad-Hoc networks with other devices that are in reach. These networks are very heterogeneous: Different hardware manufacturers provide different devices to users, equipped with different operating and middleware systems, as well as applications. Moreover, the environment to which these devices are exposed is inherently dynamic, as they are meant to be carried wherever the user goes. There have been some approaches that promote interoperability between applications running on mobile phones allowing for data synchronisation[6]; However, very few users actually update the software on their phones and mobile applications rarely react to their context and interact with each other. Industry state of the art mobile application development usually features large monolithic applications with very little code reusability.

Unfortunately, installing new applications and updating existing ones is still difficult. As a matter of fact, the only popular update is the download of ring tones and games to mobile phones. The source of the download is usually the network operator (see Figure 1(a)), and the cellular bandwidth, which is very expensive for both the user and the operator, is used for the transfer. This limited approach does little to tackle the problems of heterogeneity and context

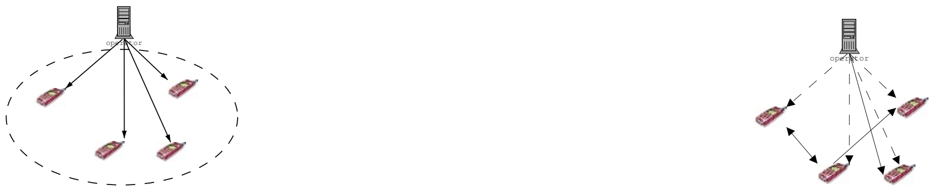


Fig. 1. (a) Deploying applications (ring tones & games) to mobile phones. Even though the phones can communicate directly using Bluetooth, they are all downloading the application from the network operator using the cellular network.(b) Possible application deployment and update: dotted lines represent security certificate download (verifying the authenticity of the code). Solid lines represent update download

awareness mentioned in Section 1. As such, the rich resources that these devices provide are not used and users experience very little context interaction. These devices are powerful enough to be used for various augmented reality applications such as interactive museum guides and tourist guides, media players which dynamically learn new codecs and interact with various services on various middleware platforms (for example a printer in a Jini system).

We believe that a better scenario would be if the devices were to form peer-to-peer networks, where each phone can use others which are currently in reach to dynamically update itself. Advertising and discovery mechanisms would allow applications to search for particular functionality from peers and download it when needed, as well as react to any change in context. Upon entering a particular building for instance, the user would be able to interact using his or her mobile phone with services offered inside the building. The network operator could be used to provide some form of digital certificate as to guarantee the authenticity of the code. Charging for this service would still be feasible, as the provider could charge for the certificate. Popular applications would be easy to locate in the peer to peer network. Less popular ones would still need to be downloaded from the network operator or another centralised service. Figure 1(b) illustrates an instance of this scenario.

Advantages of this approach include efficient use of networking bandwidth as well as automated application update & reaction to context. It constitutes the use of LM primitives (COD specifically), in a mobile environment and it requires host & functionality identification, a way to pack, request, sign and ship this functionality, an advertising & discovery service and the ability to add functionality to the middleware at runtime.

4 Principles and the SATIN Architecture

To allow adaptation and flexibility in mobile applications and to realise the scenario mentioned above, we have identified a number of principles which we have implemented in our middleware system, SATIN. We give details of these principles below.

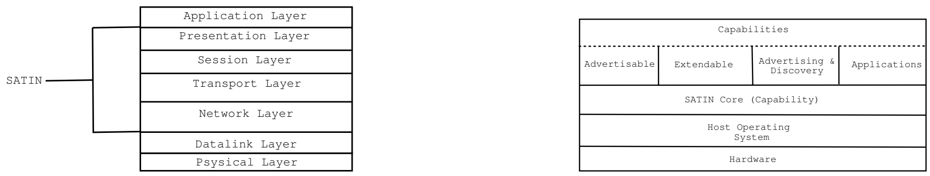


Fig. 2. (a) Our architecture in the context of the ISO/OSI networking model. SATIN capabilities represent the network, transport, session and presentation layers. (b) A high level view of SATIN. It is composed of modules, or capabilities, registered with a core.

Modularisation

For this approach to operate, we require the modularisation of both applications and the middleware system itself into a series of modules, or *capabilities*. A capability is a unit that provides a specific functionality to the user, the middleware or to other applications and adheres to a specific interface. As such, capabilities can range from a discovery technique, to a compression algorithm implementation, all the way to to a calendaring application.

The capabilities that are available to a particular host are registered with the host's registry, or *core*. The core is also a capability. An instance of SATIN is *statically* configured, if the core does not allow for the registration of new capabilities at runtime. Alternatively, it is *dynamically* configured. A reference to the handler of any capability is available to all capabilities that are registered with the core. We use a string identifier, to register capabilities with the core, which requires unique identification for each capability. As such, we do not allow for the existence of two different capabilities with the same identifier on a single host (a *locally* unique identifier). On the contrary, we propose the registration of the identifier on a centralised database, similar to the CreatorID that PalmOS[1] applications have (a *globally* unique identifier). The identifiers are thus defined by the Capability developers and checked with a centralised database to verify their uniqueness. Moreover, we allow for differentiating between revisions, or versions of each capability, by means of a version identifier. Note that implementations of the core can be distributed and not reside on the same host as the capabilities that are registered with it.

Sharply contrasted with the monolithic application development that is the current state of the art, this approach has various advantages. Considering that SATIN encourages and allows the addition of new capabilities at runtime, the identification approach allows for easily building dependency graphs of capabilities, which allows us to know whether a capability will be usable on a particular host. Moreover, it allows for identifying a capability without transmitting its interface. This decoupling approach combined with versioning allows for fine-grained application update and deployment, whereby we can update individual components of applications and libraries at runtime.

Advertising & Discovery

As our goal is to promote interoperability, adaptation and flexibility to mobile applications, it is important to be able to advertise and discover what functionality and or services are in reach. Moreover, as we are dealing with such a heterogeneous environment, it is expected that there will be many different ways to do advertising and discovery: We might use broadcast or multicast techniques, registration to and querying from a centralised server, or even interoperation with an existing middleware system, such as Jini. The modularised infrastructure described above lends itself to this, as SATIN represents different discovery and advertising techniques as different capabilities, which can be added to a host dynamically when needed.

In SATIN, different functionalities are represented by different capabilities. Capabilities which wish to advertise their presence and some information about their functionality to other hosts are termed Advertisable Capabilities (see Figure 2). All advertisable capabilities have to define a text message that will be used to describe the capability. The message is encoded in XML. For example, let us assume an FTP capability, which provides an FTP server facility that wants to be advertised. In this scenario, the capability's message might be formated as `<port>21<port><anonymous/>`. This would imply that the server is listening on port 21 and allows for anonymous access. Herein lies the importance of global identifiers: If the FTP capability is registered and has a global identifier, hosts which receive this message can decode its meaning.

This approach decouples the advertising message from the advertising and discovery mechanisms. Advertisable Capabilities can decide which Advertiser to allow advertising them. An advertiser which is allowed to advertise a capability, receives the message from it, adds some information regarding the advertisable capability itself, and then sends it over the network. For example the FTP message given above would become `<capability id=''FTP'', version=''0''> <port>21</port> <anonymous/> </capability>`. The advantages of our approach to advertising and discovery are that by representing the advertising and discovery techniques as modules, we tackle the problem of network heterogeneity and allow devices to advertise and discover functionality that is available within their current reach using various different mechanisms which can be installed and removed when needed. Moreover, by decoupling the advertising message from the advertising mechanism, we promote standardisation on describing a capability regardless of the networking medium and mechanism it is advertised and discovered with, making it easier for developers to advertise their functionality and use others that are available. Note that an advertiser can be an advertisable capability itself. This can allow devices to learn of particular advertising techniques (multicast groups for example) which are currently in reach.

Application Adaptability through Logical Mobility

Using abstractions defined by modern programming languages, we can define the following logical items that can be transferred across the network: Classes, Objects, Remote Procedure Calls (RPC) and Application Data. Note that in this context, application data may include code that cannot be directly mapped to the underlying platform (sending Python code to a Java runtime for example). We define a Logical Mobility Unit (LMU) as a combination of any of the above. In a mobile computing scenario, we add an extra dimension to the LM paradigms described above, the communication semantics. In a traditional distributed system (DS) synchronous communications is the most common communication paradigm, given the reliability of the network connection. In a mobile distributed system however, we are encountering both synchronous and asynchronous communications, with the latter being the norm, as mobile connectivity is usually fluctuating and error-prone. When tackling LM, there are more considerations that we need to take into account including how to use the received LMU, how to identify an LMU so that we are able to request it, how to formulate and structure an LMU for transfer and how to verify that the target platform is able to execute the received LMU, a problem arising from hardware and software heterogeneity. SATIN is well-suited for the construction of LMUs, because is promotes decoupling of applications into discrete modules of specified functionality, which we can identify and build dependency trees for. We represent LMUs as a container and provide mechanisms to specify the source and target hosts, the size of the unit when transfered and when deployed, an unpacker which can be used when the target host does not know how to utilise and deploy the unit received and an optional digital signature, specifying the validity of the LMU.

Recipients of LMUs can range from the core to any other capability present. Capabilities that can receive LMUs are called *extendable* capabilities (see figure 2). To use the LM mechanisms that we provide, capabilities need to use the *LM Deployment Capability* (LMDC). The LMDC is responsible for requesting, creating, sending, receiving and deploying LMUs to the appropriate extendables. The LMDC can do both synchronous and asynchronous transfer of LMUs, and allows extendables to query an LMU that is targeted at them before accepting it. A dynamically configured instance of SATIN must have an LMDC.

The advantages of this approach include that we have the flexibility of sending any logical part of an application to appropriate recipients flexibly, allowing for identification and security and for the building of dependency graphs of a particular unit. We also allow for inspection and rejection of LMUs by extendable capabilities, as well as for utilisation even if the recipient does not know how to. Our unpacker can be used to send threads around the network and our approach also allows for the arbitrary grouping of any logical parts as the situation may dictate.

5 Application Deployment through SATIN

This section describes application deployment and updating on cellular phones, using the terminology of the principles we defined on Section 4.

Let us assume a mobile phone user, with a dynamic instance of SATIN and a media player application installed, using “MPLAYER” as capability identifier. “MPLAYER” is an extended capability, as it can be updated with new codecs. The telephone is equipped with a Wavelan card. The user decides to attend a conference. At the conference, there is a computer, also running SATIN, that

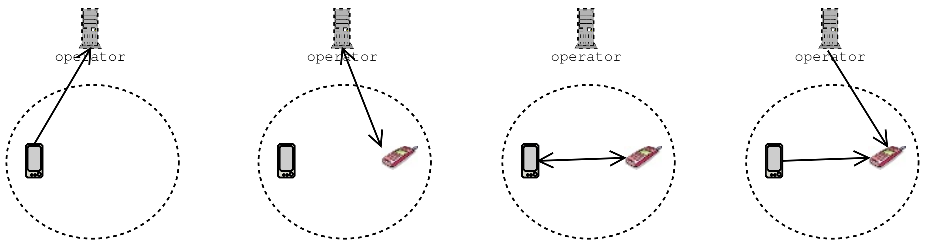
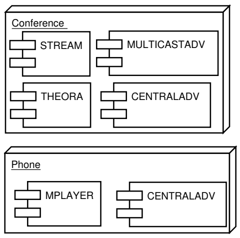


Fig. 3. (a) The conference computer's advertising service ("MULTICASTADV") registers its existence with the network operator. (b) The media player ("MPLAYER") running on the mobile phone queries the network operator for any other advertising services in the area, receives the capability "MULTICASTADV", gets charged for it and initialises it, so that it can listen to any other advertised services. (c) "MPLAYER" finds out about the stream and gets the theora codec ("THEORA" capability) from the conference computer. (d) "MPLAYER" receives a certificate for "THEORA", gets charged for it and receives the stream.

streams live videos of the presenters over Wavelan, so that attendees can get a better look of the presenter, or perhaps save the stream for later viewing, instead of taking notes. The user wishes to utilise this service, but there are the following problems: The service is advertised in a multicast group and the media player does not know which one. Moreover, the media player does not have the appropriate codec (say, theora) to display the video. We assume two advertising and discovery mechanisms, the first one registering and querying capabilities to and from a centralised host (the mobile phone operator) with identifier "CENTRALADV" and a multicast group discovery and advertising mechanism, with capability identifier "MULTICASTADV". The core's identifier is "CORE". We present a solution to this problem, using SATIN. Although some of the approaches described in section 2 could potentially be used to tackle this problem, the solutions would be too application dependent and not as flexible, forcing developers to take a number of limiting choices at design time. We illustrate the deployment of capabilities on the hosts on figure 4. Our solution is described below.

The conference's machine has an advertisable capability, "STREAM". Its advertisable message shows the location of the stream in the local network and the codec used. It only allows the "MULTICASTADV" advertiser to advertise it. "MULTICASTADV" is an advertisable capability itself, advertising the group and port it is advertising to. The "CENTRALADV" advertiser, advertises the existence of any advertisable capabilities that allow it to advertise them, to the cellular network's operator[1]. In this case, the only capability is "MULTICASTADV" which is thus registered with the cellular network's operator servers. The capabilities that "MULTICASTADV" advertises are "STREAM" and "THEORA" and their advertisable messages are periodically multicast to the group.

[1] This example assumes that the conference's computer will advertise its existence to the cellular network's operator. Obviously this is an oversimplification and done for explanation purposes



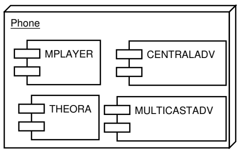


Fig. 4. (a) The deployment of capabilities before the update. (b) The deployment of capabilities on the phone, after the update.

Upon entering the conference, the user wishes to use “MPLAYER” to view the stream. He/she has the application find any available streams. “MPLAYER” queries all discovery services (“CENTRALADV” in this case), for the existence of any “STREAM” capability. It is not found and thus “MPLAYER” uses “CENTRALADV” to query for the existence of any other advertising service, currently in reach. “CENTRALADV” shows that there is an discovery service currently in reach, “MULTICASTADV”. However, the mobile phone does not have the “MULTICASTADV” capability so it requests it from the mobile phone operator using the local LMDC. The phone operator sends an LMU containing “MULTICASTADV”, a digital signature verifying the code’s validity, and specifies the “CORE” as the deployment target, charging the user accordingly. The LMU also contains an unpacker, that deploys and initialises the “MULTICASTADV” on the local core. The LMDC receives the LMU, and notifies the target (“CORE”). It accepts it, seeing that the source is the network operator and the unpacker deploys it. MPLAYER initialises “MULTICASTADV” with the information received from its advertisable message, and has it listen to the local multicast advertising group. “MULTICASTADV” notifies “MPLAYER” of the existence of the “STREAM” capability. “MPLAYER” then presents the description of the capability to the user, who decides that it is the stream he/she wants. “MPLAYER” analyses the advertising message of “STREAM” and realises that it does not have the theora codec. However, the codec, encapsulated by the advertisable capability “THEORA” is available from the conference’s computer. “MPLAYER” uses the local LMDC again to request capability “THEORA” from the computer. The computer’s LMDC packages “THEORA” into an LMU and sends it to be deployed at “MPLAYER”. The phone’s LMDC receives “THEORA” and asks the network operator to verify the validity of the code. The operator verifies this and again charges the user. The LMDC then deploys the codec to “MPLAYER” which can now display the stream.

6 Conclusion and Future Work

The use of LM techniques in traditional systems and networks is well understood and has been extensively used. As such, its advantages are well known.

The novelty of our approach, is that we provide the ability to build adaptable applications by providing the flexible use of LM techniques, using an architecture that caters for the heterogeneity that this computing scenario entails. This differs from other approaches in that we provide a complete architecture for mobile computing applications together with an engineering approach, that is specifically geared for mobile applications that can use LM techniques and we do not place any limitations in the use of those techniques. We have showed the flexibility of the approach in allowing the development of adaptable and context-aware applications, as well as dynamically deploying functionality when needed. We believe that SATIN can be used to build a new class of self-organising systems and self-healing and context-aware mobile applications. The advantages of this flexibility are many: They include improvements in ease of use, better use of limited local and peer resources, more efficient use of the network resources etc. We have implemented SATIN in Java and have been able to run preliminary tests on laptops and PDAs. The current build weighs at around 100KB, including a number of capabilities. We also have some preliminary numbers: The middleware and a sample application takes 1155KB of heap memory and transfer and installation of a single capability takes 1452ms, on an unoptimised build over Ethernet. We plan on developing more applications based on our approach that promote interoperability exhibit dynamic behaviour as well as to better evaluate performance and scalability, using a number of devices as well as simulation.

Acknowledgements. We would like to acknowledge Licia Capra for her help and comments on a draft to this paper. This work is being sponsored by EPSRC grant number R70460.

References

1. Palmsource developers program. http://www.palmsource.com/developers/.
2. K. Arnold, B. O'Sullivan, R. W. Scheifler, J. Waldo, and A. Wollrath. *The Jini[tm] Specification.* Addison-Wesley, 1999.
3. L. Capra, G. S. Blair, C. Mascolo, W. Emmerich, and P. Grace. Exploiting reflection in mobile computing middleware. *ACM SIGMOBILE Mobile Computing and Communications Review*, 1(2).
4. G. Cugola and G. Picco. Peer-to-peer for collaborative applications. In *Proceedings of International Workshop on Mobile Teamwork Support, Collocated with ICDCS'02*, July 2002.
5. A. Fuggetta, G.P. Picco, and G. Vigna. Understanding Code Mobility. *IEEE Trans. on Software Engineering*, 24(5).
6. C. Mascolo, L. Capra, S. Zachariadis, and W. Emmerich. XMIDDLE: A Data-Sharing Middleware for Mobile Computing. *Int. Journal on Personal and Wireless Communications*, April 2002.
7. Amy L. Murphy, Gian Pietro Picco, and Gruia-Catalin Roman. LIME: A Middleware for Physical and Logical Mobility. In *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS-21)*, May 2001.
8. M. Roman, F. Kon, and R. H. Campbell. Reflective middleware: From your desk to your hand. *IEEE Distributed Systems Online Journal, Special Issue on Reflective Middleware*, July 2001.

Design of a FIPA-Compliant Agent Platform for Limited Devices

Guillermo Diez-Andino Sancho, Rosa M Garcia Rioja, and Celeste Campo

Dept. of Telematic Engineering - University Carlos III of Madrid
Avd. Universidad 30 28911 Leganes (Madrid) Spain
{gdandino, rgrioja, celeste}@it.uc3m.es

Abstract. Recent advances in micro-electronic and wireless technologies have fostered the proliferation of small devices with limited communication and processing power. We think that agent technology will be of great help in pervasive systems development. Pervasive systems are inherently dynamic, with devices continuously coming and going. Agents are autonomous software entities that can interact with their environment, and therefore they adapt well to such frequent changes. However, the use of Multi-Agent Systems (MAS) in pervasive environments possess important challenges, and it is necessary to adapt their design to meet these challenges.
The first part of this paper describes the design of a FIPA compliant agent platform, adapted to the limited devices that work in pervasive environments. The second part describes the agent platform implementation in real devices, using the Java 2 Micro Edition technology.

1 Introduction

Recent advances in micro-electronic and wireless technologies have fostered the proliferation of small devices with limited communication and processing power. Personal Digital Assistants (PDAs) and mobile phones are the more visible of these kind of devices, but there are many others, embedded in the environment, unobserved. For example, today most household appliances have embedded microprocessors. The most of these devices offer a specific service to the user, but thanks to their capacity for communication, in the near future they will be able to collaborate with each other to build more complex services. In order to achieve this, devices in such "ad-hoc" networks should dynamically discover and share services between them when they are close enough. Mark Weiser in 1991 described this new age on computing as pervasive computing [1].

In this kind of environments, software applications for limited devices have to adapt themselves to the memory and processor power restrictions and to an intermittent communication with a changing quality. Even more, they need to be autonomous to reach the user's goals without continuously interacting with him. They have to be able to move around other systems to collect information or to execute tasks that device limitations do not allow performing locally or the direct connectivity does not allow to reach. The paradigm of distributed computing adapted to these characteristics is called "Agent" [2].

Our current job is based on using the mobile agent paradigm in ad-hoc networks as middleware technology to develop services in ad-hoc networks formed by limited devices which communicate directly without the need of a central system. Taking the FIPA standard as a reference, we have developed some additional features to complement the LEAP [3] mobile agent platform. The chosen technology to support the viability of our design is the Java version for limited devices, J2ME.

The paper is organised as follows: in Sect. 2 we justify and propose some simplifications to the FIPA architecture, in order to adapt it to the ad-hoc networks characteristics and the restrictions imposed by the limited systems where it will be implemented. In Sect. 3, we introduce concisely the J2ME software platform and we analyse the viability to implement the designed architecture on it, describing part of the functionality we have already developed. Finally, we enclose, in Sect. 4, the conclusions and future lines of our job.

2 Design of FIPA Compliant Agent Platform

Foundation for Intelligent Physical Agents (FIPA) started its activities in 1995 with the aim of standardising different aspects related to agents technology and multiagent systems. Nowadays, this standard can be considered as the most widely recognised and globally extended.

The reference FIPA model of agent platforms is described in *FIPA Agent Management Specification* [4]. These components are:

- Agent Management System (AMS): this manages the life cycle of agents, the local resources and the communication channels. It also provides a "white pages" service that allows agents to locate each other by name.
- Directory Facilitator (DF): this provides a "yellow pages" which identifies which agent provides what service.
- Agent Communication Channel (ACC): this manages the interchange of messages between agents on the same or different platforms. It also allows agent's migration.

A FIPA compliant agent platform should have these three components providing the interfaces and the functionality defined in their standards. In this section, we analize the possibility of simplifying the functionality of these components to do the implementation in limited devices easier and so that the offered services adapts itself to the changing environment requirements in which it is located.

2.1 Agent Management System

Agent Management System (AMS) manages the life cycle of agents, including those related to mobility, that is, it creates, destroys agents and manages the passing from one platform to another. AMS also gives support to a searcher/finder service called "white pages" which locates agents by their name.

This element is essential in an agent platform. In our current platform we keep it with full functionality as described in FIPA standard, but reducing it to a local scope in the platform. Therefore, we have eliminated the chance of extending the search to other platforms, contacting to remote platforms.

2.2 Directory Facilitator

The adaptation of the functionality of DF to limited devices operating in ad-hoc networks is one of the most important topics of research in this area. The AdHoc Working Group of FIPA focuses its current efforts on adapting the DF. In fact, the proposal described in this section is one being considered in the group [5].

In the traditional DF definition, the search of remote services is accomplished by using the concept of DF federation. DFs, besides registering services offered by local agents (native or not), they may also register other DFs (the so called federated DFs). It allows them to extend the search of services to the ones registered in these other DFs. The number of hops between federated DFs is limited by means of a parameter that restricts the depth of a search.

Now, we analyse some disadvantages of DFs federation to discover remote services in ad-hoc networks: First, the search for a concrete remote service always implies a communication with this system, due to the fact that, at the beginning, we only know one DF existing in that platform but not its registered services. That implies doing transmission with no success guaranty. Second, if search conditions allow a multihop search, i.e., a search in a remote DF may spread to other remote DFs federated in it, then it is possible that found services in those remote DFs do not be reachable from the first system, and therefore they are not valid search results.

Ad hoc Discovery Agent. To overcome the problems stated above and with the aim of maintaining the functions specified in FIPA000023 with regards to the DF, so that agents can not modify the way they interact with it, we propose the introduction of a new agent, the Ad-hoc Discovery Agent (ADA), in the FIPA architecture for ad-hoc networks. This new agent will be mandatory, and it will have the following functionality:

– To register and to update in the platform's DF, the list of remote services offered in the ad-hoc network. Therefore, the DF will have entries corresponding to remote services, not to remote (federated) DFs. Because of that the number of transmissions will be minimised.
– To propagate the search of remote services when solicited (because search conditions allow so).
– To announce services registered in the DF using the associated Service Discovery Protocol (SDP), when allowed so.

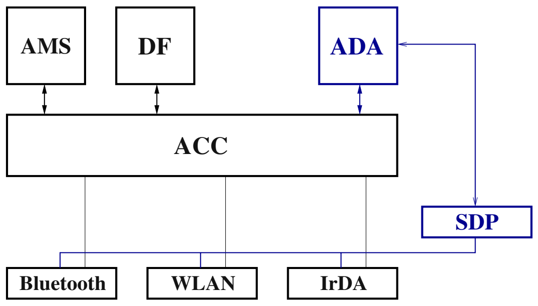


Fig. 1. Ad-hoc Discovery Agent in FIPA

ADA will use a SDP to discover remote services. In this way, a service discovery protocol adapted to the restrictions exposed in the previous section should be selected, so that an efficient solution in ad-hoc networks can be guaranteed. In [6] [7] an analysis of legacy protocols is carried out to finally propose the Pervasive Discovery Protocol (PDP), that has been defined in our working group.

The ADA provides a service that is mandatory to be registered in the DF of the platform where it resides. This way, there will always be one entry corresponding to the SDA in DFs in ad-hoc networks. When the DF processes a `search` and search propagation in the ad-hoc network is allowed (the restriction `max-depth = 1` may be reused for this) the DF will delegate the search to the ADA.

Each time the ADA (through the SDP) knows about a new remote service in the ad-hoc network, it registers the service in the DF by a `register` request. Since the network is changing, this registers will have an associated `timeout`. The timeout will be managed by the SDA, so when the time expires, the SDA will do a `deregister` request to the DF. The SDA just has to store the corresponding `agent-identifier` and its associated `timeout`.

The ADA in its turn may provide the SDP with the local services registered in the DF. To do so, the ADA will make a `search` request in the DF.

2.3 Agent Communication Channel

In a multiagent society, agents must cooperate to do their tasks and reach their goals. Traditional communication mechanisms allow two agents to transfer information, but it is not enough when the goal is reaching a social behaviour. That is, some mechanisms are required to add semantic content to the interchanged messages.

FIPA has developed ACL, which stands for Agent Communication Language. ACC in FIPA architecture is the element that gives support to communication between agents using ACL. To simplify it we have done an analisys of ACL, included below.

ACL is based on **communicatives acts**, which are in charge of passing messages, asking for information, negotiations, performing tasks and handling errors.

Sometimes conversations between agents follow some concrete patterns, repeated in occasionally few situations. Taking into account these patterns, FIPA has defined certain **protocols**. A protocol is a pattern used to guide conversations. They are guided like conversations in which each agent knows what message has to send and which ones they can receive.

Communication among Agents. The main idea of the communicative acts among agents centres around the request for carrying out an action in another agent.

When an agent A asks for an action to another concrete agent B, A sends a `request` message. The receiver can accept it (`agree` response message) or reject the invocation (`refuse` response message).

When agent A needs to do an operation but it does not know who is capable of doing it or it's known that there are several agents providing this action, a negotiation communication is established. In this case, the communication starts with a `call for proposal (cfp)` message, asking for proposals to the agents. The action to do and some conditions to take into account when the action will be performed are speficified in the request. Consulted agents send their proposals to the initial agent with `propose`, they also point the conditions of the proposal.

Agent A studies the different proposals to finally choose the best one. Then this fact is notified to the provider agent with an `accept-proposal` message and the rest of the agents are informed by with a `reject-proposal` message.

Agents can also communicate among them to interchange information. That is, agent A starts the request for information by sending a `query-if` or `query-ref` message. Later on, agent B can send the requested information (`inform`) or a failure message when a fault has occurred.

The previous explanation refers to communication among agents, but ACL is also used to ask for operations to the elements integrated in the platform, as for example AMS and DF.

Message Structure. ACL FIPA message contains one or more message elements needed to get an efficient communication among agents. It is composed of an **identifier** of the communicative act which defines the meaning of the message or the action that the emisor agent asks for with this concrete message. It also includes a sequence of **parameters** defined as a joint of couple key-value which allows to associate with each concrete communicative act all the necessary information

Minimal Group of ACL Messages. As we have seen above, ACL has a huge variety of messages types, it also has about 13 different parameters which turns the language into a so heightweigth one to be implemented in a mobile agents platform for limited devices. This requirement has obliged us to look for the minimal group of messages and its parameters so that all the necessities among agents remain covered.

When you need to request for a concrete operation you can do it with `request`, `inform` and `agree` messages. In the same way, you can use them for requesting information to the platform elements.

The negotiation process, explained before, requires lots of messages that can also be simplified. Agent A sends a `request` to the AMS or DF asking for all the agents which can perform the concrete operation. The elements of the platform send information using `inform`, and in this way the functionality given by `cfp` and `proposal` is reached. The following step consists on deciding who is required to perform the action. This task is done internally to each agent. Once it has been decided who is going to perform the operation a `request` message is sent.

To sum up, the minimal group of messages to communicate agents in a platform for limited devices is `request`, `inform` and `agree`. `Cancel` could also be replaced establishing a request time. It could also be interesting studying the compulsory minimal group of ACL parameters, that is, by using the `sender`, `receiver` and `content` parameters any action could be performed.

3 Implementation

In parallel to a FIPA compliant platform design, our working group began to evaluate the possibility of implementing this platform on real devices.

This study carried out firstly to select the implementation programming language. Nowadays we can assure that despite the developments performed in specific languages, it has been Java the most used language in the development of mobile agent platforms due to the following advantages: platform independence, secure execution, dynamic classloading, serialization mechanisms, reflection, ...

All this made us select as a base for our developments the new version of Java for limited devices J2ME, specifically MIDP. J2ME still maintains some of the characteristics making Java a good mobile agent platform programming language, but there are some other features of this language which have been restricted or even disappeared due to security issues. With regards to these changes we can find for example those allowing us to implement code mobility (dynamic class loading, serialization and reflection). We will later show on this section how we have solved these problems.

3.1 Agents Technology in J2ME

We can find several proposals trying to get involved J2ME limited devices in mobile agent platforms. The most important is the LEAP [3] project consists on a consortium of companies such as Motorola, British Telecom and Siemens that were pioneer when trying to demonstrate the viability of building agent platforms on limited devices. Its applicability is centered in networks with an infrastructure. Because of that, it does not exist a complete agent platform in mobile terminals and their operation always depends on an intermediate connected system.

Other related work has been developed in Monash University [8], in which a mobile agent platform has been developed based on the Palm Operating System.

3.2 TAgentsP: Our Mobile Agent CLDC Profile

The previous proposals have seriously contributed with regards to the adaptation of mobile agents platforms in limited devices. LEAP is a valid platform in a network with infrastructure in which it is not necessary to have an autonomous platform in the limited devices, and so it is possible to depend on a non limited system. In the scenario we want to tackle, an ad-hoc network, it is not a valid approach.

Our initial development and research is centered in providing J2ME with the dissapeared Java features that converted it into a good agent platform programming language: object serialization and dynamic classloading.

To achieve this goal and by following the J2ME modular philosophy [9], we have complemented the MIDP profile to build a new profile with the basic functionality of a mobile agent platform. We have called this profile Travel Agents Profile (TAgentsP).

Once provided the core services and with the purpose of building an autonomous mobile agent platform without the need of interacting with non limited devices we have developed a minimal HTTP server which will allows us both agent mobility and direct communication between them.

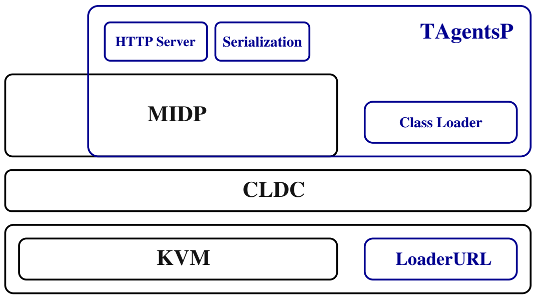


Fig. 2. TAgentsP Architecture

In Fig. 2, we can see our proposed architecture. In the next sections we will describe each of the modules we have developed and tested successfully on limited devices: the serialization module and the HTTP server.

One of the first lacks we want to solve in our new TAgentsP profile is the lack an object serialization mechanism that will allow us mobility between limited devices.

Agent mobility is achieved by converting an object into a stream of bytes travelling through the network. This stream of bytes will later be reconstructed on the destination device. This mechanism is called serialization.

In the next sections we provide a description of the developed serialization mechanism and the built HTTP server.

J2ME Serialization Mechanism. *Serialization* is a mechanism by which you can convert an object into a stream of bytes representing its state, and so on it can be transported through the network or stored in a persistent way. This conversion would not be worth if there would not be a later recovery process, which is the mechanism called *deserialization*.

The serialization is a requirement to support agents mobility due to the fact that it allows converting an agent into something transportable conserving its state. It is also a mechanism necessary when communicating messages between agents living in different platforms.

It can be said that what we are looking for is the basic support that added to the programmer's knowledge about the classes he is implementing, he can obtain a generic and extensible mechanism to convert objects into streams of bytes, being able to transport them to later recover their state on the target devices.

The main problem, to cope up with when developing serialization mechanisms, in J2ME is the impossibility to know in real time the methods and class attributes in J2ME, as we did in the standard release of Java (remember reflection).

The serialization we have carried out will make the programmer not to deal with the whole process, the data format and the serialized class recovery process. The only responsibility of the programmer will be the `Serializable` interface implementation as we show next.

The proposed solution consists on building firstly a `Serializable` interface which contains two basic methods specifying the operations any serializable class must perform: `writeObject` to serialize and `readObject` to deserialize it. The aim of this interface is to distinguish the classes that do know how to serialize/deserialize themselves.

The next step is the creation of two new classes: `ObjInputStream` and `ObjOutputStream`. These classes will control the serialization/deserialization process by performing the necessary requests to the `writeObject` and `readObject` methods every serializable class implements.

The `writeObject` method will deal with the writing of the values of the attributes of a certain class into a stream of bytes, obtaining in this way a serializable object. Using the `readObject` method this object can be interpreted allowing the construction of a new object with the information obtained from the serialized object.

If we are going to serialize a certain class, for example `Serializable_Agent` we are going to create firstly an `ObjOutputStream` object. This object will receive the object to serialize, it will build a stream of bytes in which it will write the class name of the object to serialize (this operation can be performed by calling the `Object.getClass().getName()` method supported in J2ME) to finally call the `writeObject` method of the `Serializable_Agent` class that will take care of writing the necessary information (the value of its attributes, types, . . .) into a stream of bytes. In this way it could subsequently be recovered by using the `readObject` method of the same class.

One more aspect to take into account in this project is the J2ME serializable classes library developed to help the agent's programmer. By using this library it would be easier to build serializable agents that by a mere call to the `readObject` method will be converted into a new object with the same state than the serialized object. This process will be completely transparent.

HTTP Server in Limited Devices. The TAgentsP developed profile would not be worthy if it did not exist a direct communication mechanism between mobile devices. In our development we have opted to build a minimal J2ME HTTP 1.1 server as a direct communication mechanism due to the fact that it offers an open solution which can be used for many other purposes.

Before developing the J2ME server it has been necessary to activate the serversockets in J2ME due to the fact that they are not available at MIDP level. The server has been successfully tested using a Palm with the J9 virtual machine [10]. The server can be accessed at `http://www.it.uc3m.es/gdandino/TAgentsP`.

We introduce next the server basic architecture. This architecture is composed of three main modules. The *configuration* module will define the main configuration options (number of request to accept, port, server name, supported MIME types, . . .). The *file system* module will manage the resources stored in the server. One of the problems we have found in the development of this server was the lack of an accessible file system to J2ME so that we could provide the resources requested by the HTTP clients. This module solves it by implementing a file system over RMS (J2ME persistent storage package that allows the creation and management of binary data registers) so that we can create and manage (delete, list . . .) directories and files on a J2ME device. By using the last module, *request service* all the HTTP requests (GET, POST, HEAD) will be performed.

This server can be used for many other purposes such as: class interchange between different devices to support agent mobility or direct communication between devices by using HTTP as a transport mechanism of ACL messages.

4 Conclusions and Future Works

In this paper we propose the use of mobile agent technology as a middleware to develop applications targeted to limited devices that communicate together by wireless protocols, composing the so called ad-hoc network. This is a contribution to pervasive computing due to the fact that it allows the integration of limited and embedded devices in the physical world, in distributed computing.

Taking the FIPA standard as a reference, we have carried out an analysis comprising the different defined functional elements and we have proposed a simplification so that they can be implemented in a limited device. In the case of the DF, it is not only necessary its simplification but its adaptation to the ad-hoc networks restrictions. In this way, the authors take an active part of the Working Group FIPA AdHoc that is nowadays discussing the proposals with regards to the DF, between them, the one proposed here.

Nowadays, we go on working on the design of the platform defining the interface between the ADA agent and the underlying service discovery protocol and the DF. We also go on analysing the impact the simplification of ACL could entail in the agent communication.

As we have seen, we have selected J2ME technology to implement the carried out design. We have designed and implemented a new J2ME profile, TAgentsP, which offers us the code mobility and direct communication between devices that J2ME and LEAP did not offer us. This profile has successfully been tested in real devices such as a Palm running the J9 virtual machine. Our working line is centered on the integration of our developments in the LEAP platform.

Acknowledgements.
This work has been partially supported by the Nokia University Carlos III Chair.

The authors thank their contributions to David Camacho, Carlos Garcia-Rubio and Andres Marin. Celeste Campo also thanks Florina Almenares and the members of FIPA TC/WG Adhoc, specially to Michael Berger.

References

1. Mark Weiser. The Computer for the 21st Century. *Scientific American*, September 1991.
2. H. S. Nwana. Software Agents: an overview. *Knowledge Engineering Review*, 1(3):205–244, 1996.
3. Lightweight Extensible Agent Platform. http://leap.crm-paris.com/.
4. FIPA. *FIPA Agent Management Specification*, March 2002.
5. C. Campo. Directory Facilitator and Service Discovery Agent. Technical report, FIPA Ad-hoc Technical Committee, 2002.
6. C. Campo. Service Discovery in Pervasive Multi-Agent Systems. In Tim Finin and Zakaria Maamar, editors, *AAMAS Workshop on Ubiquitous Agents on embedded, wearable, and mobile agents*, Bologna, Italy, July 2002.
7. WG-AdHoc. Agents in Ad Hoc Environments. A Whitepaper, 2002.
8. Patrik Mihailescu and Elizabeth A. Kendall. MAE: A Mobile Agent Platform for Building Wireless M-Commerce Applications. In *8th ECOOP Workshop on Mobile Object Systems: Agent Applications and New Frontiers*, Spain, June 2002.
9. Enrique C. Ortiz. A Survey of J2ME Today. Technical report, Wireless Java, 2002.
10. Websphere studio device developer. http://www-3.ibm.com/software/wireless/wsdd/.

Implementation of a Mobile Agent Platform Based on Web Services

I.E. Foukarakis, A.I. Kostaridis, C.G. Biniaris, D.I. Kaklamani, and I.S. Venieris

School of Electrical and Computer Engineering
National Technical University of Athens
Fax: +30-210-7722291, Tel: +30-210-7722289
akost@esd.ece.ntua.gr

Abstract. This paper proposes a framework that allows a convenient and flexible implementation of a mobile agent platform. We present an architecture that integrates the mobile agent computing paradigm with Web services to achieve the development of a web-integrated mobile agent platform. The platform components are deployed on Apache Tomcat web servers and the implementation is based on the Apache Java SOAP library. As it will be demonstrated, the Web services model perfectly matches the demands of the architecture of a mobile agent platform providing a straightforward design and implementation of a platform agnostic Web-based mobile agent tool.

Keywords: Mobile Agents; Web Services; SOAP; XML

1 Introduction

The use of Web services on the World Wide Web is expanding rapidly, as the need for application-to-application communication and interoperability grows in distributed computing Systems. The agent based computing introduces an important new paradigm for the implementation of distributed applications in an Open and dynamically changing environment. One very interesting application for agents is Internet computing. Mobile agents elaborate the concept of mobile Code [1] constituting a flexible and dynamic structure able to roam to remote hosts and interact with them locally. Thus, in a Web- integrated agent environment, mobile agents can be launched from one web site to another, performing transactions based on the application logic. This Scenario is especially attractive nowadays with the proliferation of wireless mobile devices; a perspective User could for example launch a Special mobile agent to gather information into the web or to perform an e-commerce task, shutdown his device and reconnect after some period of time to collect the results.

Currently, available Systems that provide integration between mobile agents and Web servers follow either the approach of an agent platform unaware of the web-server with little integration with it [2,3], or of developing custom-made web servers that are also able to host agents [4]. Other research efforts [5,6] propose different mechanisms for the close integration of agent platforms in existing infrastructures of web servers based primarily on the Servlet and JavaBeans specifications [7,8].

In contrast to the latter, our implementation is based on the Web services model and particular on the SOAP protocol [9] as used on the World-Wide Web. Again, this approach departs from the currently available systems because it adds mobile agent functionalities on top of already up and running web servers. In addition, it allows us to leverage off the massive research effort which currently focuses on Web services, SOAP and XML and enables not only a convenient access to agent services via the Web, but also a straightforward implementation of the agent management system as a collection of Web services. The platform can be easily deployed on Apache Tomcat web servers [10] using the Apache SOAP library [11] and comprises a mobile agent Application Programming Interface (API) and a collection of Java Server Pages (JSPs) for its management. The platform was designed having in mind the extensibility and flexibility. Towards this goal, the platform architecture introduces a layer between the agent management System entities and the communication channel, in order to encapsulate the communication burdens related to SOAP calls providing a simpler API to the programmer.

In the following section we give an overview of the Web services programming model and the Simple Object Access Protocol (SOAP). After introducing the basic requirements of a mobile agent platform in Section 3, we describe in Section 4 the general architecture of our SOAP based agent framework, as well as the particular components it consists of. Finally, we end with some conclusions and issues related to future work.

2 The Web Services Model

Web services are collections of Operations covered in modular, self-contained applications that are accessible over a network and generally the Internet, through standardized XML messaging. With Web services, we are on the verge of a new programming model. A set of standards has been developed that gives us programming access to the application logic of the web. This application logic is accessible to clients on every platform, and in every programming language. That is, Web services facilitate interactions among platform-independent objects, which are able to access data from anywhere on the Web. Moving away from proprietary platforms, Web services rely on loose, rather than tight, couplings among Web components.

Web services combine the HTTP communication protocol and the XML data format – and thereby benefit from both worlds. At the core of the Web services model is SOAP (Simple Object Access Protocol).

2.1 SOAP

SOAP is a lightweight protocol that supports exchange of information in a decentralized, distributed environment. It is an XML based protocol for sending messages and making remote procedure calls in a distributed environment and consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses.

Using SOAP, data can be serialized without regard to any specific transport protocol like SMTP or HTTP, although the latter is typically the protocol of choice. SOAP does not itself define any application semantics such as a programming model or implementation specific semantics but rather defines a simple mechanism for expressing application semantics by providing a modular packaging model and mechanisms for encoding data within modules. This allows SOAP to be used in a large variety of Systems, ranging from messaging Systems to RPC. A major design goal for SOAP is simplicity and extensibility. Regarding communication styles, the more important feature of SOAP is the Remote Procedure Call (RPC). With SOAP-RPC an object situated on a remote host invokes a method and gets the result when the methods returns. SOAP also provides a document or message oriented style, where the parameter can be any XML document and the return can be anything. This feature though is affected with much programming work and is rarely used.

3 Agent Infrastructure Requirements

The agent infrastructure must provide runtime support to agents for mobility and communication services and for this reason appropriate Operations must be available in the agent's implementation language. This includes operations to enable the movement of the agent to another server, to access the information space and to communicate intermediate results or questions back to their owner. Mobile agents can be implemented in a number of programming languages, but during the last years Java has proved to be the most preferable one, due to its multithreaded nature and the serialization and networking capabilities that it offers.

Mobile agents are Software entities, which can autonomously migrate from one host to another during their execution. Agents consist of code, as well as persistent state. The persistent state allows an agent to take up its work after moving to another host, at the point where it left Off before its migration. Regarding Java, the Java Virtual Machine is unable to save the execution stack of a thread. For this reason, in the event of an agent migration only the class bytecodes and the instance's data can be transported to the remote host without the execution Stack. To circumvent this problem, agents generally have to explicitly save necessary state information to member variables, allowing the entry point method to examine these variables and proceed depending on the state that the computation is in [12]. This scheme is referred as "weak migration" in contrast to the "strong migration" scheme where the transfer of the execution stack permits the infrastructure to re-instantiate the agent exactly where it executed the "move" command on a previous host.

The third part of a mobile agent consists of the attributes, which describe the agent, its requirements and history to the infrastructure. This includes data such as a unique agent identifier, the agent's owner as a target address for intermediate results, error messages about an agent's misbehavior, the place and time of origin and a movement history. This information can be tailored according to the origin or purpose of the agent in question. The advantage of this scheme is that no server needs complete information about all the possible destinations for mobile agents.

Finally, when realizing an agent platform, a name service should be used to find the location of an object. Any sensible name service for mobile agents has to use efficient mechanisms to keep track of the agents to reliably return the current location of an agent.

4 SOAP Agent Platform Architecture

To meet the previously discussed requirements and to achieve the integration of mobile agents into Web servers, we came up with the architecture depicted in Figure 1.

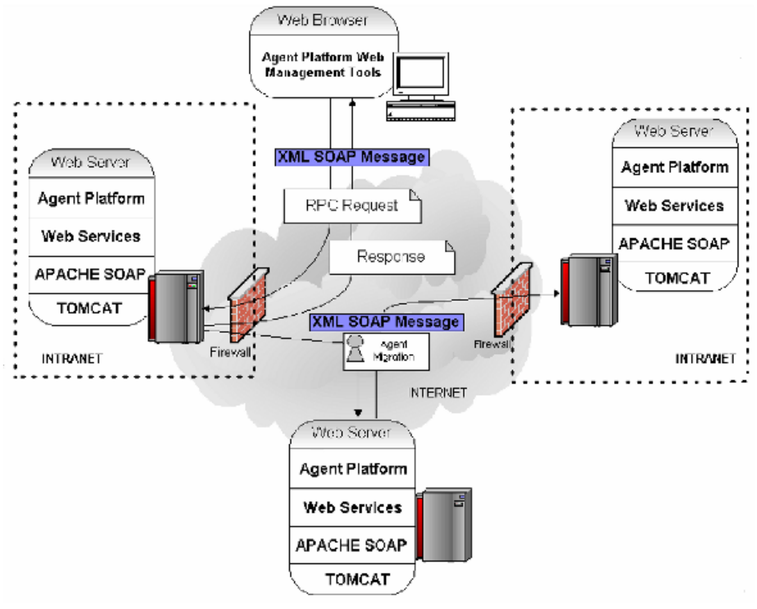


Fig. 1. The general architecture of the SOAP agent platform

The platform components were implemented entirely using the Java language and consist of the server-side core agent platform components and the client-side web management tools both deployed in Tomcat servlet containers. The platform takes advantage of SOAP-RPC using the Apache Java SOAP library. SOAP also enables the installation of the agent platform in intranets secured by firewalls. The agent platform consists of two parts, the server side and the client side.

4.1 The Server Side

Server-side contains the core platform components. As shown in Figure 2, we have adopted the MASIF specification [13] concepts of places, agencies and regions that are also used by various existing agent platforms [14]. The main difference is that now the agency and region are Web services that expose methods to other platform components.

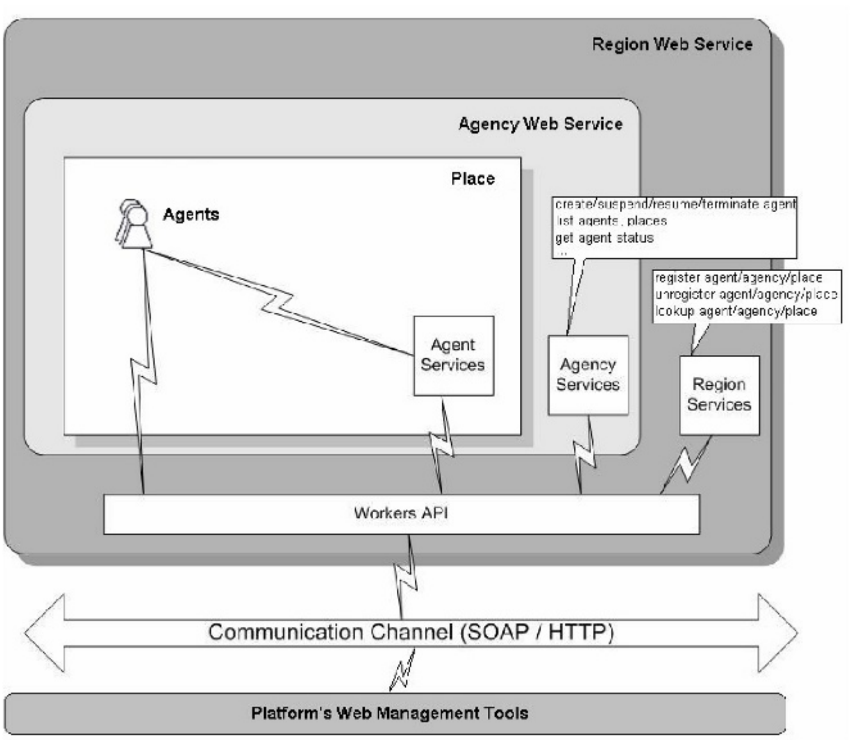

Region Web Service
Agency Web Service
Place
Agents
Agent Services
create/suspend/resume/terminate agent
list agents, places
get agent status
...
Agency Services
register agent/agency/place
unregister agent/agency/place
lookup agent/agency/place
Region Services
Workers API
Communication Channel (SOAP / HTTP)
Platform's Web Management Tools

Fig. 2. Agent platform components

Every web Container contains two main objects, namely Agency and Region, where the latter may be inactive (undeployed). The current platform implementation permits only one agency per web server.

The Region web service is used for providing directory services. Upon creation, movement, end of execution and other major events in an agent's life cycle, the region object is notified by the agencies that are registered to it, so that it keeps track of all changes. In addition, an agency or an agent (through its agency) can query the region in order to retrieve information about other agency availability, the location of other agents with use of special filters etc.

Agency web services are the core components of the platform, providing to the agents an environment to run and a communication gateway to other agencies and/or agents. The Agency web service provides the actual runtime environment for both Mobile and Stationary Agents. The Agency's main goal is to be able to create, store, move agents and allow them to communicate. Each agency is divided into different places. A place groups the functionality within an agency, encapsulating certain capabilities and restrictions for visiting agents helping this way the administrator and the programmer to separate different agents.

Calls to SOAP are being forwarded to the objects, which are responsible for handling the requests. Our main concern though, was to make the core component small and efficient, and at the Same time provide a powerful extensibility mechanism that allows higher functionality to be connected to it. For this reason we introduced the Workers API as a middle-tier between the agent platform and the communication channel. Workers are simple classes that are responsible for specific tasks, such as requesting the creation of a new Place in an Agency or the notification of the Region about changes in the status of an agent.

There are two categories of workers: those used for Agency-to-Agency communication and those used for Agency-to-Region and vice versa communication. To be able to perform its specific action, each worker must be given a number of parameters. By using these parameters, he forms the SOAP request, then forwards it to the recipients address and retrieves the result, if any.

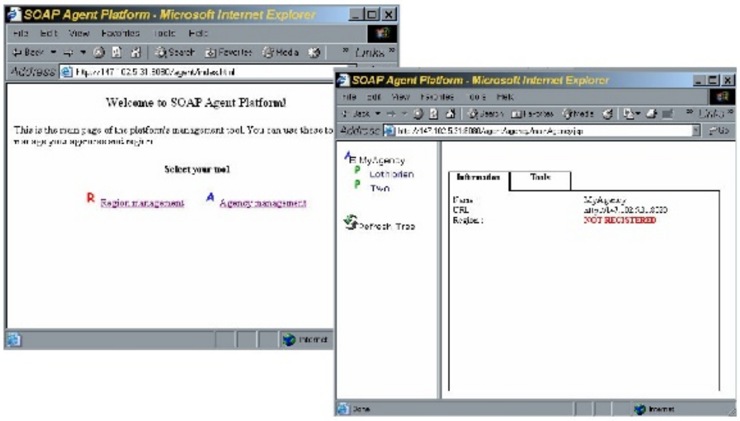
Workers are a major part of the platform. This API makes the communication between the remote objects transparent, thus making the platform easily adapted to another mean of communication (like RMI, sockets and CORBA). Another major advantage of workers is that its existence maximizes Code reusability. The client side uses heavily this API in order to assist the management of the platform.

4.2 The Client Side

The client side of the SOAP platform is controlled by several JSP pages, which permit the remote management of Regions, Agencies and agents. The management services allow the monitoring and control of agents and Places of an agency by Users. It is possible, among others, to create, remove, suspend and resume agents, services, and Places, in order to get information about specific agents and services, to list all agents residing in a specific Place, and to list all Places of an Agency. Again, the embedded code in these pages accesses methods of the Region and Agencies through the Workers API.

4.3 Communication Mechanisms

For the communication between agents the exchanges of SOAP messages follow three different schemes. The first one is simple synchronous calls. This schema


SOAP Agent Platform - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media Links
Address
Select your tool
Region management
Agency management
Internet
SOAP Agent Platform - Microsoft Internet Explorer
MyAgency
Lothlorien
Two
Refresh Tree
Information
Tools
Done

Welcome to SOAP Agent Platform!

This is the main page of the platform's management tool. You can use these to manage your agencies and region

Name	MyAgency
URL	[illegible]
Region :	NOT REGISTERED

Fig. 3. Sample JSP pages for the management of the agent platform

directly makes a call to a remote agent and returns with the result when it is retrieved. The main advantage of this mechanism is that after a call is made, the result is always ready, but the agent is blocked until the result arrives. The second method used is the asynchronous call. The agent forms a call in a matter similar to the synchronous call, with the difference that the agent's execution is not blocked this time. Afterwards, the agent can check periodically if the result has arrived. Lastly, the call-back scheme where the agent not only makes the call, but he also declares an action to be performed when he retrieves the result. The agent is not blocked this time, and its execution flow continues normally. When the result arrives, the declared action is executed.

From the programmer's perspective, it has to be noted that in contrast to stand-alone applications, where an object is passed as a parameter by reference (actually a copy of the reference is passed), with SOAP, all parameters are copied and passed with the SOAP call to the server and any modifications made to the parameters are never returned to the client. This behavior is similar with that of the Java Remote Method Invocation API.

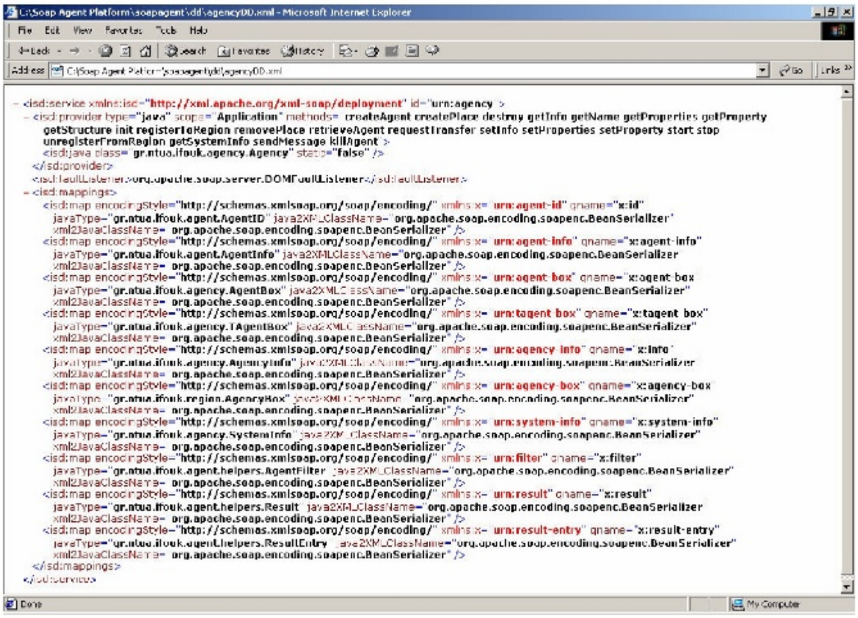
4.4 Mobility

Mobile agents in this platform are divided in two main categories: the mobile and the stationary agents. While the latter of the two remains at the agency that it was created until the end of its execution, the first one has the ability to transport itself (or ordered to be transferred) to a remote agency. This transportation is being performed by the two Agencies: the Agency in which agent currently resides in and the Agency in which the transportation targets. The two Agencies negotiate if they are able to make the transfer, and then the actual transfer is made. Each agent (like every program) consists of two parts: the actual instruction to the machine (i.e. the assembly instructions or the Java bytecodes), and the current “thumbprint” of the program in the machine’s memory. Both these two parts are transferred to the target Agency, and are assembled together in order to re-form the agent.

4.5 Deployment

As it has been previously mentioned, the realization of our Web integrated agent platform relies on the open source Apache SOAP Project and the Tomcat web server and servlet engine.

According to the Apache SOAP architecture a Web service can register itself to SOAP with an XML file called Deployment Descriptor. The deployment descriptor is an XML file that tells Apache SOAP everything it needs to properly dispatch an incoming SOAP request to the Java class that is responsible for handling it, in our case the Agency and Region classes. Elements such as the Uniform Resource Name (URN) of the service, the methods exported parameters and encoding d e s are described. The realization of the Agency and Region services requires the deployment of two descriptor files. In Figure 4, the descriptor file of the Agency Service is shown.


C:\Soap Agent Platform\soapagent\dd\agencyDD.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address C:\Soap Agent Platform\soapagent\dd\agencyDD.xml
Go Links
Done
My Computer

```
- <isd:service xmlns:isd="http://xml.apache.org/xml-soap/deployment" id="urn:agency">
  - <isd:provider type="java" scope="Application" methods="createAgent createPlace destroy getInfo getName getProperties getProperty
      getStructure init registerToRegion removePlace retrieveAgent requestTransfer setInfo setProperties setProperty start stop
      unregisterFromRegion getSystemInfo sendMessage killAgent">
      <isd:java class="gr.ntua.ifouk.agency.Agency" static="false" />
    </isd:provider>
    <isd:faultListener>org.apache.soap.server.DOMFaultListener</isd:faultListener>
  - <isd:mappings>
      <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:x="urn:agent-id" qname="x:id"
        javaType="gr.ntua.ifouk.agent.AgentID" java2XMLClassName="org.apache.soap.encoding.soapenc.BeanSerializer"
        xml2JavaClassName="org.apache.soap.encoding.soapenc.BeanSerializer" />
      <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:x="urn:agent-info" qname="x:agent-info"
        javaType="gr.ntua.ifouk.agent.AgentInfo" java2XMLClassName="org.apache.soap.encoding.soapenc.BeanSerializer"
        xml2JavaClassName="org.apache.soap.encoding.soapenc.BeanSerializer" />
      <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:x="urn:agent-box" qname="x:agent-box"
        javaType="gr.ntua.ifouk.agency.AgentBox" java2XMLClassName="org.apache.soap.encoding.soapenc.BeanSerializer"
        xml2JavaClassName="org.apache.soap.encoding.soapenc.BeanSerializer" />
      <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:x="urn:tagent-box" qname="x:tagent-box"
        javaType="gr.ntua.ifouk.agency.TAgentBox" java2XMLClassName="org.apache.soap.encoding.soapenc.BeanSerializer"
        xml2JavaClassName="org.apache.soap.encoding.soapenc.BeanSerializer" />
      <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:x="urn:agency-info" qname="x:info"
        javaType="gr.ntua.ifouk.agency.AgencyInfo" java2XMLClassName="org.apache.soap.encoding.soapenc.BeanSerializer"
        xml2JavaClassName="org.apache.soap.encoding.soapenc.BeanSerializer" />
      <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:x="urn:agency-box" qname="x:agency-box"
        javaType="gr.ntua.ifouk.region.AgencyBox" java2XMLClassName="org.apache.soap.encoding.soapenc.BeanSerializer"
        xml2JavaClassName="org.apache.soap.encoding.soapenc.BeanSerializer" />
      <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:x="urn:system-info" qname="x:system-info"
        javaType="gr.ntua.ifouk.agency.SystemInfo" java2XMLClassName="org.apache.soap.encoding.soapenc.BeanSerializer"
        xml2JavaClassName="org.apache.soap.encoding.soapenc.BeanSerializer" />
      <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:x="urn:filter" qname="x:filter"
        javaType="gr.ntua.ifouk.agent.helpers.AgentFilter" java2XMLClassName="org.apache.soap.encoding.soapenc.BeanSerializer"
        xml2JavaClassName="org.apache.soap.encoding.soapenc.BeanSerializer" />
      <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:x="urn:result" qname="x:result"
        javaType="gr.ntua.ifouk.agent.helpers.Result" java2XMLClassName="org.apache.soap.encoding.soapenc.BeanSerializer"
        xml2JavaClassName="org.apache.soap.encoding.soapenc.BeanSerializer" />
      <isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:x="urn:result-entry" qname="x:result-entry"
        javaType="gr.ntua.ifouk.agent.helpers.ResultEntry" java2XMLClassName="org.apache.soap.encoding.soapenc.BeanSerializer"
        xml2JavaClassName="org.apache.soap.encoding.soapenc.BeanSerializer" />
    </isd:mappings>
  </isd:service>
```

Fig. 4. The Agency deployment descriptor file

A major part of this XML file deals with the type mappings of several classes that the Agency service handles. Since method parameters and return values must be marshaled between client and server, the SOAP implementation must know how to serialize them. Apache SOAP uses type mappings to determine how Java datatypes should be marshaled to/unmarshaled from XML so that they can be transmitted on/received from the wire. The type mappings are stored in a type mapping registry, with the default registry being "org.apache.soap.encoding.SOAPMappingRegistry".

Each type mapping carries several pieces of information: a Uniform Resource Identifier (URI) describing the encoding style, a qualified name (qname) for the XML element, the Java class to be encoded from/decoded to, the name of the Java class to act as the serializer, and the name of the Java class to act as the deserializer. The Java classes acting as serializers/deserializers must implement "org.apache.soap.util.xml.Serializer" and "org.apache.soap.util.xml.Deserializer", respectively.

Apache SOAP though, comes with a Java Bean Serializer/Deserializer that can be used to serialize and deserialize JavaBeans using the SOAP encoding style. The public properties of the bean eventually become named elements in the XML form. For this reason the implementation of all the Region and Agency associated classes where implemented as JavaBeans.

5 Conclusions and Future Work

In this paper a Web integrated mobile agent architecture was presented that is based on the Web services model and SOAP. The Web Services Model matches perfectly the architectural requirements of a Web based mobile agent platform exploiting the existing infrastructure of Web servers as well as the massive research effort that currently focuses on Web services, SOAP and XML. The main features of the architecture are:

- Could be installed in already set-up and running web containers supporting SOAP
- Design pattern based on an intermediate "workers" layer handling the SOAP-RPC, facilitating Code re-suability, modularity and extensibility
- Web interface for management, enabling remote management of the components (Agencies and Region)
- Could be used to add intelligence to existing networks
- Can take advantage of web-servers inactivity and use computational power that was going to be wasted
- Ease of development and deployment

Our group is currently focusing on possible expansions/extensions to the agent platform such as:

- Addition of Secure Socket Layer (SSL) for agency-agency or agency-region communication and/or use HTTPS and HTTP authentication for more security
- Integration of known security algorithms (i.e. cryptography keys) to prevent execution of "evil" agents

- Addition of agent persistence. This can be achieved either by organizing a directory and file structure or by using a database.
- Addition of an agent priority schema
- Conformance to the SOAP next generation Apache Axis [15] project

References

1. A. Fuggetta, G. Picco, G. Vigna, "Understanding Code Mobility", IEEE Transactions on Software Engineering, 24(5), May 1998.
2. C. Dharap, M. Freeman, "Information Agents for Automated Browsing", in *Proc. of the ACM CIKM'96*, Rockville, USA, 1996.
3. V. Roth, M. Jalali, R. Hartmann and C. Roland, "An Application of Mobile Agents as Personal Assistants in Electronic Commerce", in *Prc.* 5th *Conference on the Practical Application of Intelligent Agents and Multi- Agents (PAAM'2000)*, Manchester, UK, April 2000.
4. G. Neumann, "High-level Design and Architecture of an HTTP-Based Infrastructure for Web Applications", in the World Wide Web Journal, vol. 3(1), 2000.
5. Paolo Marques, Raul Fonseca, Paulo Simões, Luis Silva, João G. Silva, "A Component-Based Approach for Integrating Mobile Agents Into the Existing Web Inhastructure", *IEEE Symposium on Applications and the Internet (SAINT)*, Nara City, Nara, Japan, January 28–February 01, 2002, pp. 100–109.
6. Fünfrocken S., "How to Integrate Mobile Agents into Web Servers", *Proc. WETICE'97 Workshop on Collaborative Agents in Distributed Web Applications*, Boston, MA, June 18–20, 1997, pp. 94–99.
7. Sun Microsystems Inc., "The Servlet Specification 2.3", http://www.javasoft.com/servlet.
8. Sun Microsystems Inc, "JavaBeans Specification 1.0l", http://www.javasoft.co,/beans.
9. W3C consortium, "The Simple Object Access Protocol", http://www.w3.org/TR/SOAP/
10. The Apache Consortium, "The Jacarta Project", http://jakarta.apache.org/tomcat/index.html
11. The Apache Consortium, "Apache SOAP", http://ws.apache.org/soap/
12. Iakovos Venieris, Fabrizio Zizza, and Thomas Magedanz, "Object Oriented Software Technologies in Telecommunications, From theory to Practice", John Wiley & Sons, April 2000.
13. OMG MASIF, "Mobile Agent System Interoperability Facility (MASIF) specification", ftp://ftp.omn.org/pub/docs/orbos/97-10-05.pdf.
14. IKV++ Technologies, "Grasshopper Agent platform", http://www.grasshopper.de/
15. The Apache Consortium, "The Axis project", http://ws.apache.org/axis/

EasiShop: Enabling uCommerce through Intelligent Mobile Agent Technologies

Stephen Keegan and Gregory O'Hare

Department of Computer Science, University College Dublin,
Belfield, Dublin 4, Ireland.
{Stephen.Keegan, Gregory.OHare}@ucd.ie

Abstract. Emerging *uCommerce* technologies offer new means and methods of discovering, brokering and delivering a seamless transaction mechanism, providing convenient and efficient retailing possibilities for both merchant and consumer. Within this paper we introduce a set of interoperating technologies to enable an automated *context-sensitive* uCommerce system using mobile agent interaction facilitated by the Bluetooth wireless transmission medium. We describe an architecture and a working prototype - EasiShop - based on these ideas and illustrate the operation of the prototype in an example scenario.

1 Introduction

This paper describes the operation, design and development of Easishop, an agent-based, context-aware uCommerce shopping application. Easishop seeks:

- To partially automate the shopping process, thus alleviating the user from what may be a difficult, time consuming or unpleasant task;
- To provide a means by which the shopper can be informed of optimum consumer conditions–cheapest price, best availability, best after sales service, geographic proximity–all with minimal effort on the part of the shopper;
- To deliver a system architecture that facilitates *multiple retailers and shoppers*, interacting and competing in a dynamic, self-contained *macro-economy,* thereby benefiting the shopper.

2 Related Research

The use of agent technologies within the ubiquitous and mobile sector in and of itself is not new. Several other projects have sought to effectively deploy such approaches. Exemplar systems include The Impulse project [1] which commissions the Hive Agent System. Similarly, the Agora project [2] provides a mobile shopping framework implemented through the Zeus Agent Framework. Other notable systems include a mobile agent m-commerce framework, as proposed by Mihailsecu and

Binder [3] which provides for different agent types - *Device Agent, Service Agents* and *Service Providers*.

3 The Easishop User Experience

The Easishop system is initiated when the user provides the system with a set of personal and preferential information, from which a basic user profile is constructed. This information includes such details as gender, age, financial parameters and a requested level of system intrusiveness. The information is input by the user using a standard *Personal Digital Assistant* (PDA). When the basic user profile has been input by the user, it is processed and retained in the system. At this point, the user is ready to specify to the system which products or class of products are sought or which might be of interest. Again, this input is achieved using a standard PDA input mechanism via a custom-built *Graphical User Interface* (GUI). Figure 1 illustrates.

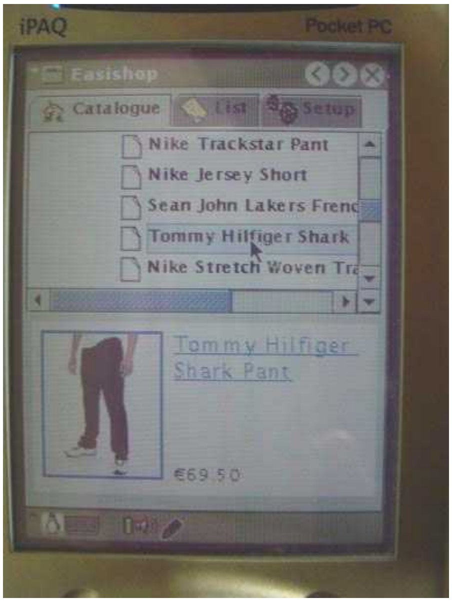


Fig. 1. The Easishop Graphical User Interface (GUI)

The user is thus equipped with a PDA containing on-board user shopping preferences - the *shopping list* - as well as a basic user profile. The system becomes functional as the user walks down a high street equipped with the PDA. This PDA also hosts an EasiShop *PDA Agent* – an encapsulated semi-autonomous software entity that is charged with the primary task of acquiring items on the shopping list, in accord with user preferences (e.g. lowest price, most suitable colour or closest location). To implement the Easishop system infrastructure, it is envisaged that each participating retail outlet will incorporate an *EasiShop Hotspot (EH)*, a wireless broadcast zone, sourced from a device sited within or adjacent to the shop entrance. The EHS constitutes an area within which automated interaction may take place between the

PDA (acting on behalf of the user) and the store. Each EHS incorporates a network connection to a centralised server – the *Easishop Marketplace*. It is here that *Store Agents*, representing a multitude of relevant retail outlets, compete for the user's custom.

Upon walking past an Easishop-enabled retail outlet, the user enters it's EH. At this point, the PDA Agent is kindled and communicates with the *Storefront Agent*, inquiring as to whether this outlet stocks items which are on the user's shopping list or items or items which may be of interest to the user. Before the stage at which the PDA is due to negotiate with the user, a *Shopper Agent* is transferred from the PDA to the retail outlet's store server. Here negotiation takes place. If negotiation is unsuccessful, further migration ensues, with representative agents from the PDA and the retailer component both migrating to the Marketplace. This scenario is shown in figure 2 below.

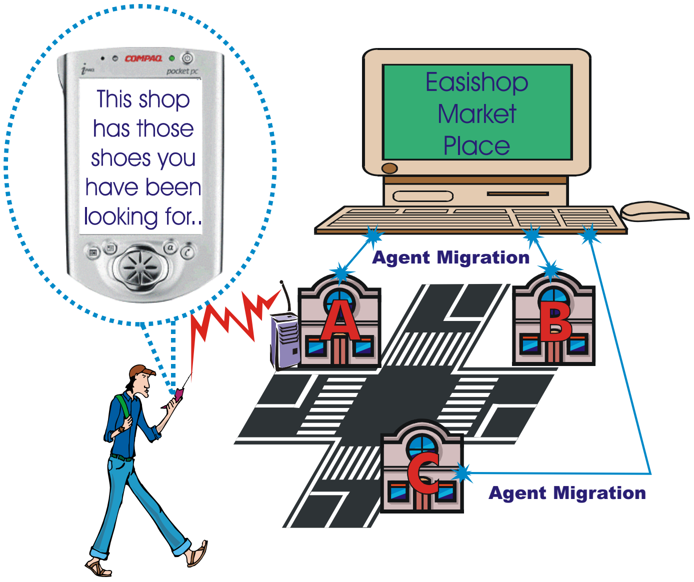


Fig. 2. The Easishop Scenario

The Marketplace provides a forum where shoppers and retailers alike can interact with impunity from the movements of the individual shopper. At the Easishop Marketplace, a *stall* is created for this user's agent and for a number of interested *Store Agents* (including the agent representing the store that the user has just encountered) who are geographically relevant and who stock each product that is sought by the user. It is important at this stage to note that the user's current location can easily be determined as the EHS of the active outlet. This could give rise to enhanced competitiveness - this store may deem the conversion of this potential customer as being of heightened importance. Conversely, competing outlets may attempt to lure the user away from the EHS of the competitor with added enticements. Store Agents compete for the Shopper Agents' business whilst Shopper Agents compete to secure desired products at offer prices. In this way, a macro-economy emerges based upon user location and competitive market forces between different retailers.

Once a purchase has been fulfilled by the Shopper Agent, the agent passes the credit card details to a special *Secure Transaction Agent* (which also resides at the Marketplace), in the process authorising the payment to the order of the agreed amount. At this point the Shopper Agent will attempt to remigrate back to the user's PDA. In the event that the user has left the active EH, the Shopper Agent remains at the Marketplace until such a time that remigration is possible. When remigration occurs, the PDA interrogates the Shopper Agent, before informing the user of details of the recent purchase.

4 Design and Implementation

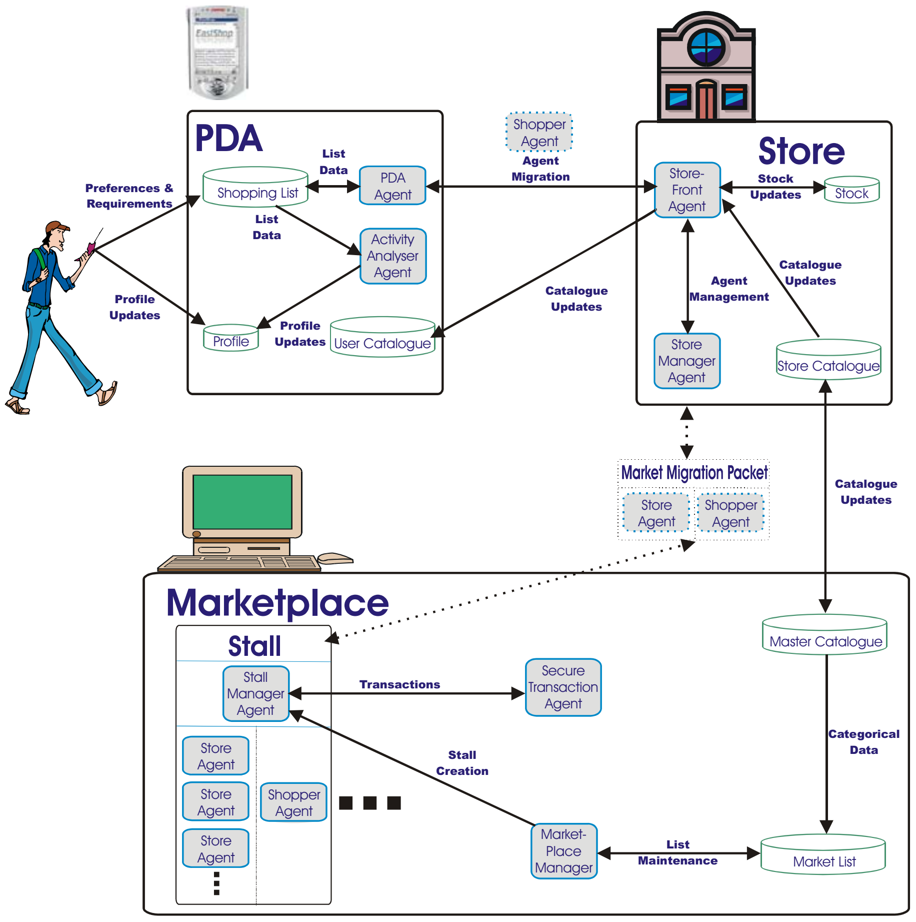


Fig. 3. Easishop System Architecture

Easishop adopts a classic client-server architecture. The PDA components compose the client-side implementation, providing a solution whereby lightweight processing (mostly graphical and non-intensive data-manipulation routines) is performed client-side. Emphasis is placed server-side with the Marketplace assuming the role of principal component of the Easishop architecture, at least from an agent-oriented point of view. It is here that the bulk of the inter-agent negotiation, financial settlement and migrational management are carried out. The Easishop store server can be regarded as a conduit between client and server. Figure 3 describes this structure.

The agents depicted in the preceding figure are divided into 2 distinct categories – *environmental agents* and *system agents.* The *MarketPlace Manager Agent* and the *Store Manager Agent* are of the environmental agent category. They are not created on a one-to-one basis. Instead, the store server may contain a number of Store Manager Agents, just as the marketplace may instantiate more than one Maketplace Manager Agents. In other words, environmental agents may be created and terminated according to changing environmental conditions such as system usage. This approach ensures that the agent is *inherently scalable*. System agents encompass all the other agents in the system – i.e. PDA Agent, *Activity Analyser Agent,* Storefront Agent*,* Store Agent, Shopper Agent*, Stall Manager Agent* and the *Secure Transaction Agent*. The relevant locations of environmental and system agents are depicted in the Easishop layered architecture in figure 4.

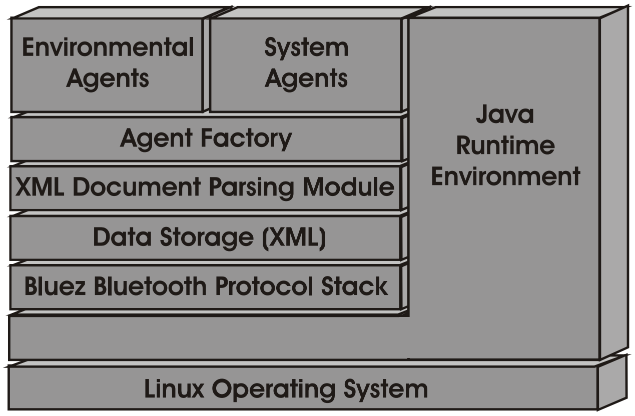


Fig. 4. Easishop Layered Architecture

4.1 PDA Agents

The primary agent housed on the user's PDA is the *PDA Agent*. This agent forms the rational machinery of the Easishop client. The agent's principal concern is to scan for Easishop Hotspots. When an EHS is detected, the Storefront Agent is queried to determine if the outlet is of relevance to the user's requirements. The PDA agent refers to the shopping list as well as the user profile to determine if this is indeed the case. If the outlet is of sufficient interest to the PDA agent, then this agent fabricates a Shopper Agent, which reflects the needs and predilections of the user that is destined to migrate to the Easishop Marketplace.

The PDA Agent, in attempting to establish the user's requirements, refers to the products on the user's shopping list. It also refers to the user profile in an effort to determine what product interests may be implied therein. The composite set of prerequisites constructed from this pairing is used to stipulate the belief set of the Shopper Agent, which in turn is utilised as the negotiation dataset with the storefront agent. Of secondary concern to the PDA agent is the necessitated *User Catalogue* update mechanism. This ensures that the on-board User Catalogue, stored as an XML file is up-to-date with current product description, image data and pricing data. The PDA Agent manages this mechanism, ensuring that, during inter-agent communication with the storefront agent, updates are received from the *Master Catalogue*. The primary function of the Activity Analyser Agent is to build a user profile, based upon the user's behaviour, which will enable the delivery of content which is more relevant and more targeted toward the individual user's preferences. The user's behavioural patterns are used to infer preferences. A simple Bayesian classifier [4] creates a user profile from the user's preferences. The profile can be used to calculate the probability that any product (or product classification) is interesting to the user.

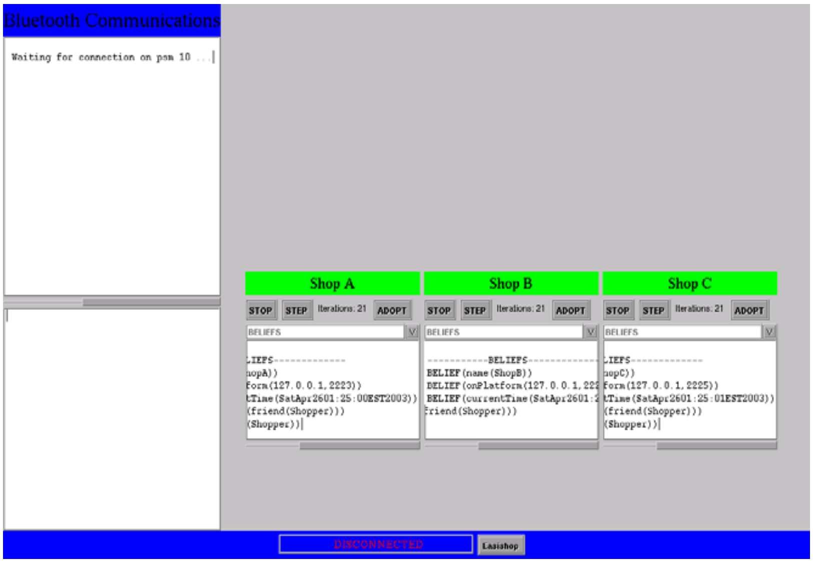
4.2 Store Agents

The function of the Store Manager Agent is to co-ordinate the communication dynamic between the retail outlet and the user. It accomplishes this through the creation, maintenance and disposal of Storefront Agents. The Storefront Agent is charged with essential liaison with the PDA Agent. When a PDA Agent enters the Easishop Hotspot, the Storefront Agent is queried as to what kind of products are on offer. The Storefront Agent responds with a presentable list. If the PDA Agent returns positive, indicating that it is indeed interested in certain product(s), the Storefront agent initialises a migration window, whereby the Shopper Agent, which has been pre-fabricated on the PDA, is paired with a representative agent from the outlet. (Store Agent). Together, these agents comprise a *Market Migration Packet*, and are promptly transferred to the Marketplace.

4.3 Marketplace Agents

Upon reception at the Marketplace of the Market Migration Packet, the Marketplace Manager Agent publishes the details of what product is involved in a data sink known as the *Market List*, a collection of data indicating all currently active markets, viewable by all agents. A *stall* is created at this point and all interested agents are permitted to enter the auction process. A standard *English Auction* [5] ensues, though the system is sufficiently generic to permit instantiations of other auction types. [6].

Figure 5 shows three active Store Agents, while figure 6 shows an auction in progress after a Shopper Agent has been admitted to the stall.



```
Waiting for connection on psm 10 ...
```

Shop A — Iterations: 21

```
LIEFS-------------
nopA))
form(127.0.0.1,2223))
tTime(SatApr2601:25:00EST2003))
(friend(Shopper)))
(Shopper))
```

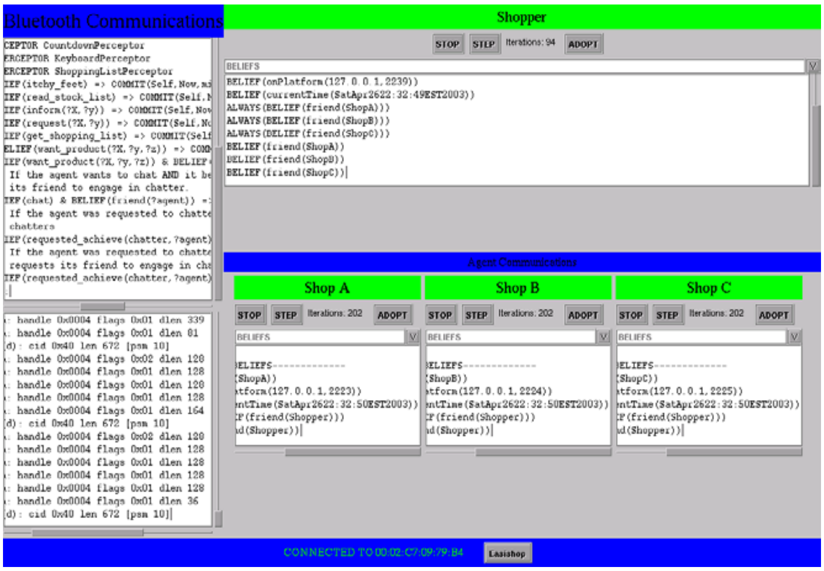
Shop B — Iterations: 21

```
----------BELIEFS-----------
BELIEF(name(ShopB))
BELIEF(onPlatform(127.0.0.1,222
BELIEF(currentTime(SatApr2601:2
friend(Shopper)))
```

Shop C — Iterations: 21

```
LIEFS-------------
nopC))
form(127.0.0.1,2225))
tTime(SatApr2601:25:01EST2003))
(friend(Shopper)))
(Shopper))
```

Fig. 5. The Marketplace containing a stall with three Store Agents



```
CEPTOR CountdownPerceptor
ERCEPTOR KeyboardPerceptor
ERCEPTOR ShoppingListPerceptor
IEF(itchy_feet) => COMMIT(Self,Now,mi
IEF(read_stock_list) => COMMIT(Self,N
IEF(inform(?X,?y)) => COMMIT(Self,Nov
IEF(request(?X,?y)) => COMMIT(Self,Nc
IEF(get_shopping_list) => COMMIT(Self
ELIEF(want_product(?X,?y,?z)) => COMM
IEF(want_product(?X,?y,?z)) & BELIEF(
 If the agent wants to chat AND it be
 its friend to engage in chatter.
IEF(chat) & BELIEF(friend(?agent)) =:
 If the agent was requested to chatte
 chatters
IEF(requested_achieve(chatter,?agent)
 If the agent was requested to chatte
 requests its friend to engage in cha
IEF(requested_achieve(chatter,?agent)
-|
```

```
: handle 0x0004 flags 0x01 dlen 339
: handle 0x0004 flags 0x01 dlen 81
(d): cid 0x40 len 672 [psm 10]
: handle 0x0004 flags 0x02 dlen 120
: handle 0x0004 flags 0x01 dlen 128
: handle 0x0004 flags 0x01 dlen 120
: handle 0x0004 flags 0x01 dlen 128
: handle 0x0004 flags 0x01 dlen 164
(d): cid 0x40 len 672 [psm 10]
: handle 0x0004 flags 0x02 dlen 120
: handle 0x0004 flags 0x01 dlen 128
: handle 0x0004 flags 0x01 dlen 128
: handle 0x0004 flags 0x01 dlen 128
: handle 0x0004 flags 0x01 dlen 128
: handle 0x0004 flags 0x01 dlen 36
(d): cid 0x40 len 672 [psm 10]
```

Shopper — Iterations: 94

```
BELIEF(onPlatform(127.0.0.1,2239))
BELIEF(currentTime(SatApr2622:32:49EST2003))
ALWAYS(BELIEF(friend(ShopA)))
ALWAYS(BELIEF(friend(ShopB)))
ALWAYS(BELIEF(friend(ShopC)))
BELIEF(friend(ShopA))
BELIEF(friend(ShopB))
BELIEF(friend(ShopC))
```

Shop A — Iterations: 202

```
ELIEFS-------------
(ShopA))
tform(127.0.0.1,2223))
ntTime(SatApr2622:32:50EST2003))
F(friend(Shopper)))
d(Shopper))
```

Shop B — Iterations: 202

```
ELIEFS-------------
(ShopB))
tform(127.0.0.1,2224))
ntTime(SatApr2622:32:50EST2003))
F(friend(Shopper)))
d(Shopper))
```

Shop C — Iterations: 202

```
ELIEFS-------------
(ShopC))
tform(127.0.0.1,2225))
ntTime(SatApr2622:32:50EST2003))
F(friend(Shopper)))
d(Shopper))
```

Fig. 6. The same stall after a Shopper Agent has migrated. At this point an auction is occurring

4.4 XML Document Parsing Module

Many of the currently available java-based XML parsers (e.g GDome, Docuverse), are unsuitable for Easishop due to relatively large memory footprint and associated processing overhead. Furthermore, it was seen as desirable to implement a customised solution in light of scalability and enhancement capabilities.

4.5 Data Storage (XML)

XML [7] is well suited to uCommerce in general and to Easishop in particular. The language is specifically designed to suit a heterogeneous and dynamic environment such as Easishop. XML is likely to emerge as the *lingua franca* of uCommerce and allows scope for expansion and optimisation that the system demands. As previously discussed, the Easishop system agents negotiate *pro rata* in an attempt to locate and acquire appropriate products on behalf of the user. Hence the development of a suitable product description schema *data description syntax* is imperative for the success of the Easishop construct.

Furthermore a standardised product database is required as a means of allowing users to articulate their shopping needs and to enable the coordination of those needs with the Easishop system agents. A number of initiatives have been undertaken in an attempt to classify products in a standardised, coherent and logical manner. The United Nations Standard Product and Services Classification (UNSPSC) [8] was seen to be suitable for the needs of EasiShop, since it encodes a global market of 12,000 product codes and constitutes an open global standard. To create the main product database, an XML version of the complete UNSPSC scheme was construed. The resultant XML tree was customised for usage within the Easishop system. This included tags that would enable cost details, summarised and detailed product descriptions as well as embedded visual components – i.e. an associated jpeg image for each individual product.

4.6 Agent Factory

Agent Factory [9] is a cohesive environment that delivers structured support for the development and deployment of agent-oriented applications. Specifically, Agent Factory supports the fabrication of a type of software agent that is autonomous, situated, socially able, intentional, rational and mobile. This is achieved through the formulation of a multi-layered architecture that provides a high-level agent programming language based upon a logical model of commitment, a distributed run-time environment that delivers support for the deployment of agent-oriented applications, an integrated toolkit that delivers a visually intuitive suite of tools and a software engineering methodology that specifies the sequence of steps required to develop and deploy agent-oriented applications. All Easishop agents are delivered through Agent Factory.

4.7 BlueZ Bluetooth Protocol Stack

BlueZ is the official Linux Bluetooth protocol stack. It is an Open Source project distributed under GNU General Public License (GPL). To facilitate Easishop, a module that accesses and utilizes the BlueZ core was developed. It then became necessary to construct an interface to permit information flow between the BlueZ module and the JRE.

4.8 Java Runtime Environment (JRE)

The Blackdown JRE 1.3.1 RC1 for the Linux/ARM architecture is utilized on the PDA. Kaffe, a clean room implementation of the Java virtual machine, plus the associated class libraries needed to provide a JRE is used at the Store and Marketplace hosts. Blackdown is the JRE of choice for implementation on the PDA since it offers sophisticated GUI functionality. This includes the Swing components like JTree and JEditPanel, which allow for complex XML document display and manipulation techniques.

4.9 Linux OS

The Server-side components reside on a linux 2.4.18-3 server. The PDA (an Ipaq 3870) runs the *familiar v0.6.1* linux build (2.4.18-3) from handhelds.org. Many of the inherent benefits of linux are applicable to the Easishop implementation. These are widely documented and include stability, security, open source, network orientation, speed and efficiency.

5 Conclusions

This paper presented the operation, design and development of Easishop, an agent-based, context-aware uCommerce shopping application. Easishop, in contrast to other agent-based uCommerce applications, provides strong, intentional and mobile agents. System assessment involved conducting a series of performance and scalability tests. The former specifically measured the reliability and efficiency of agent migration, results indicate that agent migration takes on average 320ms. Scalability tests have investigated the performance degradation as the agent community increased. The results of these tests are depicted in figure 7.
Detailed usability tests are ongoing.

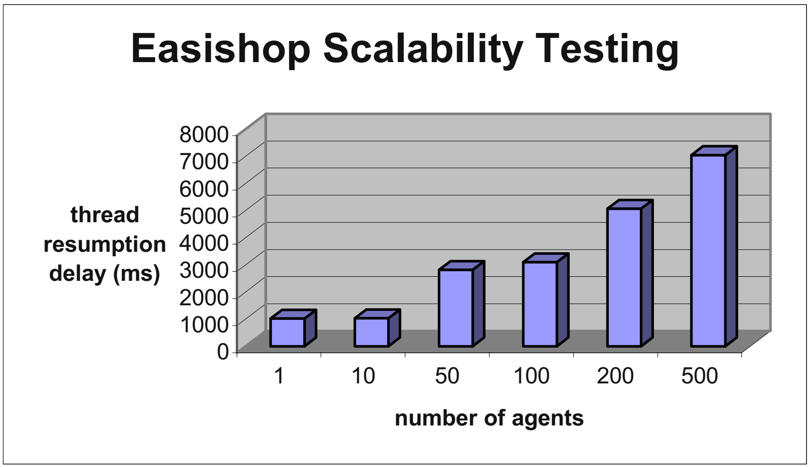


Fig. 7. Easishop scalability testing. [1]

References

1. Youll J., Morris J., Krikorian R., Maes P.: Impulse: Location-based Agent Assistance, Software Demos, Proceedings of the Fourth International Conference on Autonomous Agents (Agents 2000), Barcelona, Spain (2000)
2. Fonesca S., Griss M., Letsinger R.: An Agent-Mediated E-Commerce Environment for the Mobile Shopper. Hewlett Packard Technical Report HPL-2001-157 (2001)
3. Mihailescu P., Binder W.: A Mobile Agent Framework for M-Commerce, Proceedings of Agents in E-Business (AgEB'01), Vienna, Austria (2001)
4. http://kmi.open.ac.uk/projects/bkd/
5. http://www.fipa.org/specs/fipa00031/
6. http://www.agorics.com/Library/auctions.html
7. W3C (1999b). Extensible Markup Language. In http://www.w3.org/XML/
8. http://www.unspsc.org
9. O'Hare, G.M.P., Jennings, N.R. eds. 1996. The Agent Factory: An Environment for the Fabrication of Distributed Artificial Systems. Foundations of Distributed Artificial Intelligence, Sixth Generation Computer Series, Wiley Interscience Publishers, New York, pp 449-484.

[1] Tests performed using a Dell Dimension XPS T500. Specification: 1 x 400 MHz Pentium II (512 KB L2 cache). 128 MB RAM (PC100 ECC SDRAM). 9 GB 7200 rpm Hard Disk (Western Digital Enterprise. Testing script: java -ms64m -mx128m Easishop (JIT disabled))

Agent Migration over FIPA ACL Messages

Joan Ametller, Sergi Robles, and Joan Borrell

Computer Science Dept. Universitat Autònoma de Barcelona
08193 Bellaterra, Spain `Joan.Ametller@uab.es`

Abstract. In this paper, we present the design and implementation of an inter-platform mobility (migration) mechanism for agents. This migration is based on FIPA ACL messages. We also evaluate the performance of this implementation Agent mobility is an essential requirement for some electronic commerce applications, such as those in the Sea-of-Data (SOD) family. The majority of agent systems at present do not support mobility or do not respect standards. This situation does not encourage inter-operability. Our implementation provides a Mobile Agent System that is compatible with the FIPA standard, and SOD applications can be implemented using it. Our solution has been confirmed as feasible by performance evaluation, even in situations of extreme agent load. Already existing applications may make use of our idea to increase their functionality and flexibility, as our results have been integrated in the well-known JADE agent platform.

1 Introduction

Sea-of-Data (SOD) applications are those in which the user is interested in the results of processing large amounts of data in several distributed locations. These data may not leave their location for a number of reasons, such as legal requirements, bandwidth limitation, or restrictions due to the acquisition process (the data may be acquired on demand). The user's connection may be intermittent, subject to sudden cuts or irregular, either because data processing requires a lot of time or because the user is connected via an unstable network, or from a simple device in a pervasive computing environment.

These applications are extremely useful in areas such as intrusion detection systems, medical image processing or satellite image analysis. They also provide new perspectives for electronic commerce, as this data processing may be sold to clients as an indirect service.

Because of the characteristics and restrictions of SOD applications, it is difficult, or almost impossible, to implement them using traditional technologies. Mobile agent systems appear to be the most feasible solution for their implementation. In these systems, autonomous software entities (agents) may move across a network of execution platforms (platforms). The applications code can thereby be executed wherever the data are located, without the need for centralizing all the processing. Furthermore, the user, once the mobile agent has been sent, can remain disconnected. The use of agents also enables the execution of the application to be parallelized.

There are two basic requirements for developing electronic commerce applications in the Sea-of-Data field. The first essential requirement is an agent system that is inter-operable with other agent systems, or the equivalent, an agent system that meets FIPA specifications.

FIPA (the Foundation for Intelligent Physical Agents) [5] has so far been the main body that has promoted interoperability between agents. It provides the specifications for a standard language for agents (*Agent Communication Language - ACL*), ontologies structuring the semantic contents of these messages, and an abstract platform model.

Of the existing agent systems (for a complete review, see for example the doctoral thesis [10]), one of the few that meets FIPA specifications is JADE (Java Agent DEvelopement Framework) [8]. It is also easily adaptable to the needs of SOD applications because it is open source. JADE is currently the agent system used as a basic platform by most groups carrying out research and development in the field of agents [9].

The second essential requirement, obviously, is to provide the chosen agent system with mobility, in order to have an interoperable mobile agent system with open source. This second requirement is not an immediate task.

Firstly, the FIPA specifications have not gone into sufficient depth in the field of mobile agents. The only initiative has been to propose a specification for agent mobility of an optional nature and with little precision [4]. This specification has become obsolete because it has not been implemented by any agent system. Secondly, previous experiences, such as the development of the MARISM-A platform [3], show that there are various possible ways to provide agents with mobility (based on sockets, Remote Method Invocation, etc.), but few of them meet the basic FIPA interoperability specifications.

In the case of JADE, this platform includes a mechanism for transporting agents internally within a single platform by means of introducing the concept of distributed agent containers. This approach, as analyzed in this article, is not valid for constructing a mobile agent system between different platforms.

In this article, we present a proposal for providing inter-platform mobility (migration) to the JADE platform based on the ACL language, the basic mechanism for communication between FIPA agents. This proposal has been let known to the managers at JADE and the FIPA technical committee. This proposal will enable us to have an open system for mobile agents, in which a number of applications can be implemented, including Sea-of-Data and electronic commerce related.

The second and third sections concentrate on presenting the main characteristics of FIPA and JADE, respectively. In the fourth section we describe our proposal for agent migration based on ACL. The fifth section includes the evaluation of our proposal's performance. Finally, the sixth section is given over to conclusions and suggestions for continuing our work.

2 FIPA

FIPA is a non-profit making organization that was established in 1996 with the aim of promoting specifications for facilitating interoperability between agent systems. The model proposed by FIPA consists of concrete specifications enabling interoperability, based on a high-level abstract architecture.

FIPA's abstract architecture [6] defines all components that should be found in an agent platform as high-level entities. Some of these components are optional, and others are compulsory. The FIPA specifications are a concrete production of this architecture and are basically focused on two points.

Firstly, the basic elements which the platform should consist of are defined, as well as some aspects of internal high-level functioning such as the states that form the life cycle of the agents. Three components are defined - AMS, DF and ACC. AMS (Agent Management System) is an agent which any new agent in the platform must locate in order to send it a registration request when its execution starts. This agent is mandatory in all FIPA platforms. This agent thereby maintains a register of descriptions of all agents in the platform that may be consulted by any agent, thus providing a directory service. The DF (Directory Facilitator) is an optional component which may or may not be represented by an agent, and provide agents with a Yellow Pages service. The ACC (Agent Communication Channel) is a high-level interface, through which messages are sent from one platform to another, using an MTP (Message Transport Protocol). Figure 1 shows the location of these components within FIPA platforms.

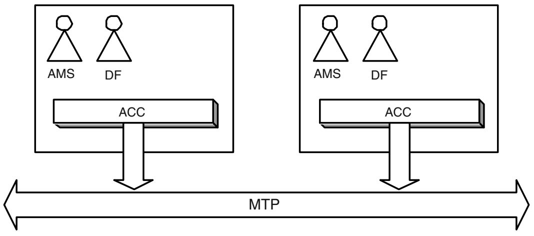


Fig. 1. FIPA Platform Model

Secondly, the specification gives minute details of the entire process and characteristics involved in the sending and structure of messages between agents (ACL). This part of the specification describes the common language used between the agents and the way concepts are represented in the messages. It also specifies the functioning of the transport system that platforms must use to send messages to other platforms. This is the equivalent of saying that an agent is capable of sending a message to another agent belonging to a different platform and that platforms are capable not only of establishing communication, but also of continuing the protocol started according to the content of the message. The level of understanding between the two agents will be greater or lesser depending on the knowledge that they have of the concepts sent in the message, but there is always a minimum level of comprehension that means that the conversation is not blocked by an unusual message.

Platforms meeting FIPA specifications are known as FIPA-compliant. There is a European project called *AgentCities* [1] which is an initiative for creating a network of FIPA-compliant platforms. Every platform offers services available to everybody thanks to the interoperability provided by FIPA.

Agent migration capacity has not been dealt with in detail by FIPA. This is because while on the one hand it is feasible to standardize a message transport layer, on the other it is not so feasible to standardize a migration between heterogeneous systems. In this last case, the system should enable agents to migrate and continue with their execution within an environment for which they have not been designed. If we also consider that the platforms may be written in different programming languages, and the security mechanism problems posed by executing remote code in a machine, the complexity of a specification of this type grows exponentially. However, some time ago a specification for migration [4] appeared that gave details of how the migration process should be carried out, sending the code of agents within ACL messages. This specification was not completed and became obsolete because it was never implemented.

3 The JADE Platform

The platform chosen for implementing migration was JADE, because it is a widely adopted platform within the software agent development and research communities. It is open source and complies with FIPA specifications. This platform facilitates its agents' mobility, but as shown below, does not meet the requirements for a general migration (inter-platform migration).

The platform is internally composed of a large number of functional modules, which can be placed into three categories in general terms:

Core. The core of the platform is formed by all components providing the necessary execution environment for agents' functioning.

Ontologies and Content Languages administration. This consists of the platform's mechanisms for carrying out information processing in ACL messages, and the internal structures that the platform and agents will use to represent this content.

Message transport mechanisms. Mechanisms and protocols used to send and receive messages at both intra-platform and inter-platform level.

At the core of the JADE platform is the concept of the container, which is the minimum execution environment necessary for an agent to operate. Each container in JADE is executed in a different Java virtual machine, but they are all interconnected by RMI (Remote Method Invocation).

Containers do not only enable groups of agents to be separated into different execution groups, but platforms may also be distributed in various machines so that each has one or several of them. One of the different existing containers is the principal, which represents the agent itself and which gives orders to all the others. JADE also provides mobility between containers. For this reason, if the platform is distributed in various machines, agents can move between them. However, accepting this type of mobility as a general migration could be considered a mistake. "Satellite" containers are highly dependent on the principal and many operations carried out by the agents within them end up passing through the central node. Furthermore, the connections between them (carried out by RMI) must be permanent, as if not, many errors due to the loss of link may be generated. As we can see, using this type of mobility as a general migration ends up making mobile agent systems' scalability disappear because a certain type of operations is centralized in a single node. However, it may be very useful to use the diagram of containers to distribute the processing of platforms that have to bear a heavy load or to isolate some platform types within a single platform for security reasons.

These details lead to the necessity for inter-platform migration, which is carried out through a non-permanent channel and makes a system of mobile agents available that is much more scalable, and in which platforms are totally independent units. This independence is not only desirable from the point of view of fault tolerance, but also because of privacy.

4 Our Proposal for Migration Using ACL

The idea of creating a migration using ACL messages came from FIPA's specification regarding mobility, where this type of migration is proposed. However, as mentioned above, this specification only gives a general outline of the ontologies, the protocol, and the life cycle of a mobile agent. It has not been updated due to the lack of implementations and has become obsolete within the FIPA specifications. For these reasons, we have found that there is a need to propose extensions to the specification to cover situations that it does not deal with. We have had some positive feedback about this extensions from people close to the JADE project and the FIPA technical committee.

The design for a migration using ACL means that transmission of mobile agents between two platforms will be carried out using the message system between agents. In other words, the agent (both the code forming it as well as the state that it is in) will travel as the content of a message between two agents. Specifically, the agent will travel in the message between the AMS agents of each of the platforms involved in a migration.

Because the platforms have mechanisms for sending and receiving messages, using a parallel transmission protocol is not necessary. This is an advantage in interoperability terms and enables agents to be transmitted using the various message transport protocols (HTTP, IIOP, SMTP, etc). Furthermore, this is achieved in a totally transparent way.

The first logical step in this process is to design the ontology and the protocol that will be used in the exchange of messages between the two platforms. This protocol has the movement of the agent as its final purpose. Defining an ontology basically consists of defining the elements that will form the content of an ACL message to give a common interface between the two parties when extracting the information of the message.

The two possible migration models that are proposed in the initial FIPA specification deal with one migration directed by the agent and another directed by the platform. In our implementation, we have decided to adopt the migration directed by the platform, which is more robust, as it enables the platform to decide which migration requests are accepted and which are not.

The ontology initially specified by FIPA is made up of seven elements: five concepts ("mobile-agent-description", "mobile-agent-profile", "mobile-agent-system", "mobile-agent-language", and "mobile-agent-os") and two actions ("Move" and "Transfer").

Of these concepts, we only use the first one, "mobile-agent-description". This is because it is very difficult to develop systems that enable agents with different architectures to migrate with total interoperability, at the current level of agent technological maturity. These agents could have been written in different languages or executed in different operating systems. For this reason, these concepts are never used, assuming that mobile agents which migrate move between the same platforms. Obviously, if the platform has been developed in a language like Java, a migration between platforms lodged in different operating systems is possible. However, this is a characteristic of this language which is transparent to the platform, and therefore does not involve the need to use the concept “mobile-agent-os”, for example. In any case, although we do not use it, we maintain these concepts to ensure compatibility in case it is possible to make agents migrate between platforms implemented in a different way or with different languages.

The concept “mobile-agent-description”, on the other hand, is highly useful to us. Within it are several fields that define the characteristics of the mobile agent in question. Among others, these include the characterization of the code and its data.

Of the two actions specified, we have only implemented “Move”, for migrations directed by the platform. We do not take the “Transfer” action into consideration, which can be used for migrations directed by the agent, although it is supervised by the platform.

Once the content of the ACL messages was described, we moved on to enumerating the protocol by which the migration process is carried out. A diagram of the messages exchanged during the migration process can be seen in figure 2.

- Firstly, the agent wishing to migrate starts a conversation with the AMS agent of the local platform to make a request for migration. This request is the first step in the standard FIPA-Request protocol and consists of a Request-type message with a Move action and a *MobileAgentDescription* (*MobileAgentDescription* is the name of the class that implements the concept of “mobile-agent-description”), in which the code and data fields are empty.
- When the AMS agent receives the request for migration from a mobile agent, the first thing it does is to decide whether to accept it or not according to a given criterion. If the migration is accepted, the AMS agent sends an *Agree* message to the agent, or if not, a *Refuse* message.
- If the migration is accepted, the first thing that the AMS agent does is to obtain the code and data (serialized instance) of the agent that made the request and fills in the code and data fields of the *MobileAgentDescription*.
- The next step is to make contact with the AMS belonging to the platform to which the mobile agent wishes to migrate. To this end, the local AMS must start a parallel conversation to that between the agent and the remote AMS.
- When the remote AMS receives the request, the agent’s code and data travel within the *MobileAgentDescription* that the local AMS has prepared.
- Following its own criteria, the remote platform decides whether to accept the incoming agent. If so, it responds with a *Agree* message, and if the agent does not meet the requirements specified by the platform to execute it, it responds with a *Refuse* message.
- The remote AMS loads the agent’s class, deserializes its instance and restores its execution. Once this entire process has been successfully completed, the standard FIPA-Request protocol is completed by sending an *Inform* message to the local AMS.

– The final step in the protocol consists of informing the agent that started the process. If the process has been successfully completed, the original agent is destroyed.

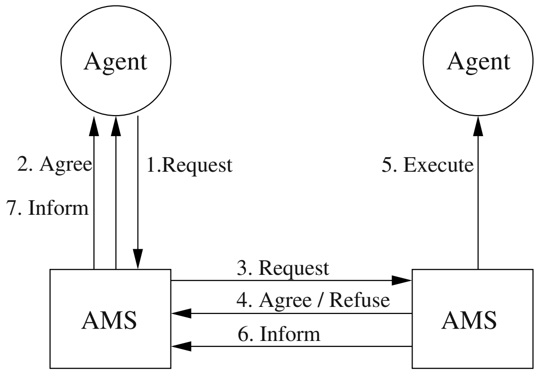


Fig. 2. Exchange of ACL messages in the migration protocol

4.1 Design Components

After describing the ontology and the protocol used, the components necessary to carry out the entire process are listed below.

Migration Manager. The Migration Manager component is responsible for carrying out all operations enabling the execution of an agent to be suspended in the local platform. Also, it enables its state and code to be processed. These operation allow to insert the agent in an ACL message. In the remote platform, it is responsible for undoing the messages that contain agents, decoding them and acting together with the platform to restore the agent's execution.

Class Loader. The class loader is the component that constructs the representation of the class in the memory so that it can be used in the code. The bytecode of the class is extracted from the ACL message and loaded during the deserialization time of an incoming agent. At the same time, each classloader provides a space of different names from the others, so that two agents created with one class with the same name do not conflict within a single platform if they are loaded by different classloaders.

Mobile Agent. We have created the *MobileAgent* class, inheriting the properties of a basic agent and adding the functionality of being able to migrate to it. This functionality provides it with the *doMigrate(dest)* method, which starts the migration protocol when invoked.

Conversation Modules. These modules were implemented using the behavious of the JADE model [2]. Behaviors represent agent tasks and are also a useful tool for designing the protocols that govern conversations using ACL. There are basically two components of this type developed to provide mobility. The first is the agent's behavior. This component is launched when the function *doMigrate(dest)* is invoked and is responsible for supervising the migration from the agent's point of view. The second was implemented within the AMS agent to help it administer its part of the migration protocol. This behavior has a double functionality as it was designed for playing the roles of the local AMS and the remote AMS. The most complex part of the implementation of this component is its functioning as a local AMS: parallel conversations (AMS-Agent and AMS-AMS) which depend on each other must be taken into consideration.

5 Performance Evaluation

The objective of performance evaluation is to confirm how the migration System implemented influences the normal functioning of an platform. The migration process involves many processes such as the serialization of agents, their encoding to Base64, the sending of large ACL messages, and dynamic class loading. The aim is to determine whether the implementation of the migration is feasible or whether it involves an acceptable decline in the performance of the platform.

We have seen how our proposal for migration of agents sent within ACL messages. The JADE platform is able to support the massive sending and receiving of messages. The main difference between a migration of this type and sending inter-platform messages lies in two areas. Firstly, the size of an ACL message containing an agent's code and data is in general significantly greater than the average size of a normal ACL message. Secondly, the pre-process involved in extracting the agents from the ACL message, as well as that of creating the message based on the agent, makes the typical treatment of a message somewhat complicated.

The trials carried out took place on machines based on Pentium IV at 2GHz with 256 Mb of RAM, a Java 2 virtual machine, and a VLAN within an Ethernet network switched to 100 Mbps, in stable and low load situations. The version of the JADE platform installed in each was 2.61.

The aim of the trials was to check the processing behavior of overloaded platforms faced with a significant number of simultaneous migrations. An agent with the test process using almost the entire computing capacity was used, to obtain the maximum decline in the platform's performance. Once the time taken for this process in a totally unloaded platform was calculated (see Row 1 of Table 1), this time was calculated again, but submitting the platform firstly to hundreds of simultaneous migrations from incoming agents, and then to hundreds of migration requests from outgoing agents. The aim of these two trials was to confirm the performance in the two facets of migration - the sending and reception of agents. Finally, the process was repeated but loading the same amount of static agents. This enabled us to compare the load of the agents with migration. The trials were carried out five times for each experiment. Table 1 shows the results for 300 agents, sized from 3,4 to 15,2 Kbytes. The processing of the line of messages in JADE, and therefore the requests for migration, was carried out using a limited number of execution threads, meaning that the processing of the sending and reception of messages was treated in an almost sequential way. The 300 migrations evaluated represent an overload situation in the transport system during a significant time period with regard to the duration of the test process.

Test/time (seconds)	Test 1	Test 2	Test 3	Test 4	Test 5
Test Process (TP)	211 s	209 s	212 s	212 s	209 s
TP + 300 incoming migrations	227 s	228 s	226 s	228 s	226 s
TP + 300 outgoing migrations	225 s	227 s	227 s	227 s	226 s
TP + 300 loads	225 s	227 s	226 s	225 s	225 s

Table 1. Performance evaluation

The results in table 1 show how the migrations carried out decrease the efficiency of the test process by an average of 15 seconds, but it can also be seen how the agents' temporary load weight (see Row 4 of Table 1) is similar to that of the migration (see Rows 2 and 3 of Table 1).

The conclusions to which the results obtained lead us are as follows. Firstly, the decline in the system's capacities during periods in which there is a massive and constant migration flow is acceptable, as this flow does not paralyze the system or markedly affect the processes carried out in the platform. Secondly, if the duration times for the test process during the migrations and during agent loading are compared, it can be deduced that most of the resources used during the migration process are due to the classloading of the agents. Classloading is not the typical migration process as although it is used in it, it also takes place when new agents are input into the platform. From this we can deduce that the procedures that are exclusive to migration (negotiation, serialization, etc.) are a load that is hardly significant.

6 Conclusions

In this paper, we have presented a proposal for the mobility of agents between various platforms, based on the agent communication language (ACL) proposed by FIPA. This has been the key to fulfilling the double objective of maintaining consistency with the generally accepted platform model and also permitting interoperability between platforms of a different type.

The implementation has been carried out as an extension of JADE, the most commonly used agent platform at present. Those responsible for the development of this platform are aware of our implementation and it could be considered to be included in future versions. During the design of the mechanism, a subsequent extension in order to guarantee the security of information has been taken into consideration. The flexibility it provides enables it to be rapidly adapted for using encoding and authentication schemes, as anticipated by the FIPA technical committee for security.

After carrying out various experiments, we have confirmed that the system's performance does not decline even when it has to administer hundreds of migrations while other agents are executed. Our migration proposal is therefore feasible for both large and small systems. Optimizing the mechanism and the natural extension of the platform model also mean it can be used in small mobile devices, such as the PDA.

There are basically three areas in which this work may be continued. Firstly, the scheme could be expanded, by adding privacy and authenticity protection mechanisms, considering the security aspects that will arise in new ACL specifications. Secondly, to improve overall performance, the Java threads planner could be modified, to include a reactive architecture [7]. This would make the system suitable for processing thousands and tens of thousands of agents. Finally, another of these lines could be the development of applications using our scheme.

Acknowledgments

This work has been partially funded by the Spanish Government Commission CICYT (TIC2000-0232-P4), and Catalan Government Department DURSI (2001SGR 00219).

References

[1] AgentCities.NET. European Commission funded 5th Framework IST project. November 2001. `http://www.agentcities.net`
[2] F. Bellifemine, G. Caire, T. Trucco, G. Rimassa. JADE Programmers Guide. July 2002.
[3] CCD Research Group: MARISM-A, An Architecture for Mobile Agents with Recursive Itinerary and Secure Migration. `http://www.marism-a.org` (2003)
[4] FIPA Agent Management Support for Mobility Specification. Foundation for Intelligent Physical Agents. Geneva - Switzerland 2000.
[5] FIPA. Foundation for Intelligent Physical Agents. `http://www.fipa.org`
[6] FIPA Abstract Architecture Specification. Foundation for Intelligent Physical Agents. Geneva - Switzerland 2002. `http://www.fipa.org`
[7] L. Hazard, J.F. Susini, F. Boussinot. “Junior Reactive Kernel”. Inria Research Report 3732, July 1999.
[8] JADE, Java Agent DEvelopment Framework. `http://jade.cselt.it`.
[9] S. Poslad, S. Willmott. Agent Engineering for Open Systems. 5th European Agent Systems Spring School. Universidad Autónoma de Barcelona - February 2003
[10] S. Robles. Mobile Agent Systems and Trust, a Combined View Toward Secure Sea-Of-Data Applications. Phdthesis, Universitat Autònoma de Barcelona, 2002.

A Mobile Agent Manager

Rui Pedro Lopes[1] and José Luis Oliveira[2]

[1] Polytechnic Institute of Bragança, ESTiG, 5300 Bragança, Portugal
`rlopes@ipb.pt`
[2] University of Aveiro, , DET, 3810 Aveiro, Portugal
`jlo@det.ua.pt`

Abstract. Friendliness on user interfaces has been mostly forgot in mobile agent research. In fact the multitude of problems that have to be solved have left this more "simple" piece of the package out of target. However, the success of technology comes also from this important layer in the product development.
This paper describes an application that integrates monitoring data from the mobile agents environment, process and presents information in different ways according to different user requirements and allows the user to remotely control mobile agents entities.

1 Introduction

A mobile agent (MA) is a piece of software that is able to migrate from a particular environment to some other remote site where it can execute a specific function (system configuration, information gathering, ...) and then migrate to other systems in order to continue its job [1].

To accomplish this scenario, mobile agents do require runtime platforms to provide them with the necessary resources they need to operate and to migrate. Besides, this support system must also handle agents' life-cycle, security issues and, some times, additional services like multi-agent communication or coordination. This infrastructure is known as the agent system (AS) or agency, according to a well accepted nomenclature, and it is developed as an operating system extension.

From a more conceptual view, the typical role of a mobile agent is to perform an action in the representation of some entity, person, service or agent. The definition of its role is typically embedded in the agent code but its itinerary can be statically or dynamically defined by the owner/user or, using its own initiative and according to collected data, it can infer the following movement. During the mobile agents navigation there are several problems that can arise related to communication, access control, agents rights, agencies accessibility, just to name a few. In this context, the user should be able to monitor and control what they are doing, individually or as a community, and to change its behavior whenever it is necessary.

One of the problems with current mobile agent systems is the considerable difficulty in maintaining a large, distributed and remote infrastructure. The absence of standards for agent systems APIs (Application Programming Interfaces) limits or even eliminates the possibility for interoperability between different vendors' products. This also means that the tools available for monitoring and management are proprietary. However, it is important to monitor and control the infrastructure to conduct performance evaluation and performance tunning, fault recovery/diagnostics, early detection of error or lapses on the overall service and load-balancing [2].

As agents work all over the network, the user must know what is going on globally and what is the impact on the network and on the agencies. He should be provided with information such as:

- What is the navigation plan of an agent? From where did it come and to where is it going?
- What is it doing?
- Where are the agencies geographically located?
- Which one is the most visited agency on the network?
- What is the CPU usage on every agency?
- How much time does this agent spends in each agency?
- What is the relation between the agency load and the number of mobile agents currently visiting it?

The user interface provided by some agent system (AS) is usually based on the File Manager paradigm which is insufficient to transmit an adequate and rich idea about the state of the mobile agents as individuals or as a community, the relations among them and the impact that they are causing in the network resources. Such paradigm is useful for structuring static information and for accessing hierarchical structures that do not change frequently, such as region information – the mobile agents directory service, agent systems and places –, but they cannot cope with several and very dynamic parameters. It may be adequate to represent the entities inside the AS, however how could this tree-like view show agents appearing and disappearing as they are created and destroyed? How could it show where migrations happens the most? Moreover, migrations would not show up until the user opens a branch (by clicking on the '+' sign for instance). It is also very difficult, if not impossible, to represent past and future information (where did this mobile agent come from? Where is it going to?).

The research around MA have not taken to much attention on the user interface. In fact, most of the past problems come from technical operational constraints and from the search for adequate application scenarios. Even the agent systems interoperability has not been too important on the last years research. However, mobile agents already have a role in the computing and communication market and the credibility of this technology will increase as solutions for interoperability, security assurance and friendly management will appear.

This paper presents a visual-oriented manager that helps to control mobile agents execution and navigation across multiples agencies and regions.

In order to present information in an intuitive user interface it is necessary to retrieve data from the agent system. Good sampling strategies and a well planned information system are necessary to avoid overcharge of the network bandwidth and computational resources. This subject will be detailed in section 2. In section 3 we describe a graphical user interface that can represent geographical information together with MAs and agent systems working parameters. Its main characteristic is the possibility to show, in an integrated view, the overall system operation. Finally, in section 4 we present some implementation details, followed by some conclusions.

2 Mobile Agents Information System

The information system behind the operation of a mobile agent infrastructure is structured in two main parts. One is related to the services provided by the agents, such as the operations they perform, the itinerary they have or their current state; the other is related to the parameters that represent behavior, such as "where they are" or "what are they doing". From the user perspective, the former is meaningful for defining mobile agent based operations and the later is useful for infrastructure management purposes.

When agents are migrating and therefore changing their location repeatedly, the user may have more difficulty in maintaining an updated and accurate knowledge about where they are and what they are doing. The task of monitoring a distributed environment is more complex than the task of monitoring a single host or application. The number of AS can easily rise to dozens or hundreds and the performance of the systems depends on several factors including the network throughput and topology. It is important not to overload the network with the sampling mechanism but it is also important to avoid loosing resolution by using an insufficient sampling frequency [3].

The information system should be as generic as possible to allow using a broad set of mobile agent systems, regardless of the technology and vendor. The Mobile Agent Facilities (MAF) of the Object Management Group (OMG) provides such generalization and is a good starting point [4].

2.1 Information Type

The user has access to the mobile agent infrastructure by retrieving and sending information. The information allows him to start operations as well as to check on how things are going. To design the information system we started the user requirements by enumerating the operations that should be available to the user.

The first condition was the integration of facilities of agent systems coming from different vendors. The user should be able to start jobs or tasks with specific paths or services. This means that the user should be able to create mobile agents and to provide them with appropriate arguments – even if the operational constraints of an agent framework avoids its agents to run in different systems (it will be confined, in this case, to the agencies of the same kind). He should also be able to monitor the agent location and to suspend, resume or terminate the agent operation.

Tracing and changing the agent's path in real time is also an important feature. This allows the user to keep a record of past and foreseen movements. To maintain this information accurate, it is necessary a) some notification mechanism or b) continuous polling on the agent location. The later may require large bandwidth to work properly. If the number of migrations per second between two agencies, which we call the connectivity between agencies, is high, this means that both agencies play an important role for the operation being performed by the mobile agents or that something is wrong with the paths they should be walking.

There are also some other useful parameters to evaluate the behavior of the agents. These are the CPU load of the host where the agencies are located, the number of mobile agents currently sharing an agency and the number of static agents. These values should be possible to correlate, to make it possible to take decisions on the capacity of the host to support the number of agents to the available processing power. It is also important to store historical information about these parameters so that we can present to the user their evolution and to compare past with current values.

MAF is a collection of definitions and interfaces that provides generic access to mobile agent systems. It uses OMG IDL to declare two interfaces, which define all the operations on the agent system and on the region: the `MAFAgentSystem` and the

`MAFFinder`. They can be used to provide vendor and platform independence [5]. Table 1 presents a summary of the information system as well as the available methods to get/set the information.

Table 1. Summary of the operations associated with the information system

User Operation	Available method
Create agents	MAFAgentSystem::create_agent()
Monitor agent location	MAFFinder::lookup_agent()
Suspend, resume agents	MAFAgentSystem::suspend_agent()
Resume agents	MAFAgentSystem::resume_agent()
Terminate agents	MAFAgentSystem::terminate_agent()
Trace agent movement	Callback, notification or event mechanism (Proprietary)
Explicitly change the agent path	The user must change the agent state and desire (Proprietary)
Agency connectivity	Results of the ratio between migration tracing and time
CPU load	Operating system query (Proprietary)
Number of MA in the agency	MAFAgentSystem::list_all_agents()
Number of static agents	MAFAgentSystem::list_all_agents()

In addition to changing values, there are also several sampling operations that must be performed periodically to build the information system. The next section focuses on this subject in grater detail.

2.2 Sampling and Changing

For the agent infrastructure to be contacted, either for retrieving values or for changing parameters, it is necessary to provide a common mechanism that both parties share. Moreover, it was decided that the mechanism should be independent of the agent system technology and vendor, perform automatic measurements and provide a minimal impact on network and computational resources. A further desired feature is that the sampling could be performed periodically or on demand based on a specific event. This requires time stamping every measurement so that it can be appropriately correlated with past and with related measurements.

The communication depends on two characteristics: the syntax – the way information is transported, and the semantics – the meaning of the information. Agent system communication is usually performed through an API which allows remote, high level access to the system parameters. This API is seldom standard but there are several standard protocols and models that can be used, such as SNMP (Simple Network Management Protocol) [6][7], CORBA [4] or other, since the agent system supports them (Fig. 1).

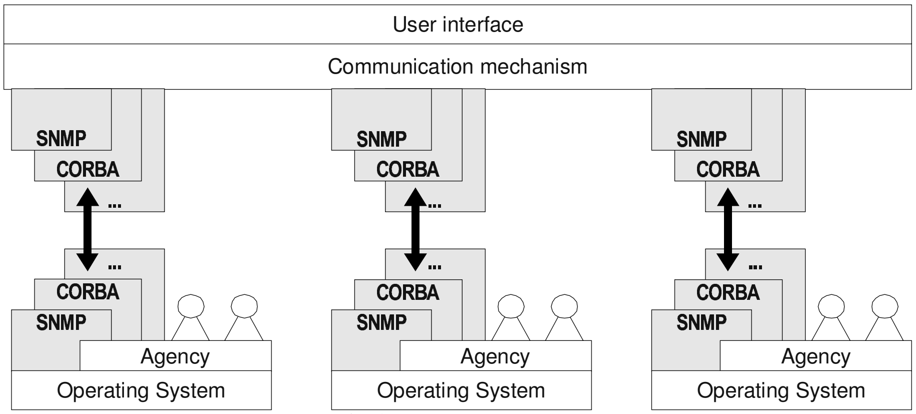


Fig. 1. Sampling mechanism

The communication mechanism must follow the syntax defined by the agent system. Semantics has to do with the meaning of the information and implies some kind of processing. Some parameters are usually more meaningful if processed before being consumed. For example, the number of migrations between two agencies is not meaningful unless associated with time information. Saying that 2312 migrations occurred between two agencies is not meaningful until we say "in the last 20 seconds". Other parameters may be meaningful by themselves, such as processor load or the number of mobile agents currently parked at an agency. This makes us foresee two kinds of sampling: absolute – the sampling value corresponds to the retrieved value – and delta – the sampling value corresponds to the difference between the current and the previous values.

The information is presented to the user in a single location, so it has to travel there at least once. However, centralized sampling may have some scalability problems, related to the network latency, overhead and resource usage.

After having the sampling values, sometimes it is necessary to perform some kind of processing to correlate them or to further enhance its meaning. This requires processor power and may also represent another scalability problem.

To cope with these problems, we suggest a distributed sampling mechanism based on mobile/static agents [8]. This should be seen as a dynamic tree of delegated sample processors. Their responsibility is threefold:

- Resolve syntax problems by defining a gateway between the management application and the agent system;
- Maintain a log of previous samples to provide historical information and to allow the remote processing of information;
- Perform basic processing operation on sample values.

The latter is seen as semantic compression because it extracts meaning of the raw data thus reducing the amount of information transmitted to the management console. To further cope with latency and to help saving network resources, the distributed samplers perform data compression by applying a lossless compression algorithm to the information before transmitting it (Fig. 2).

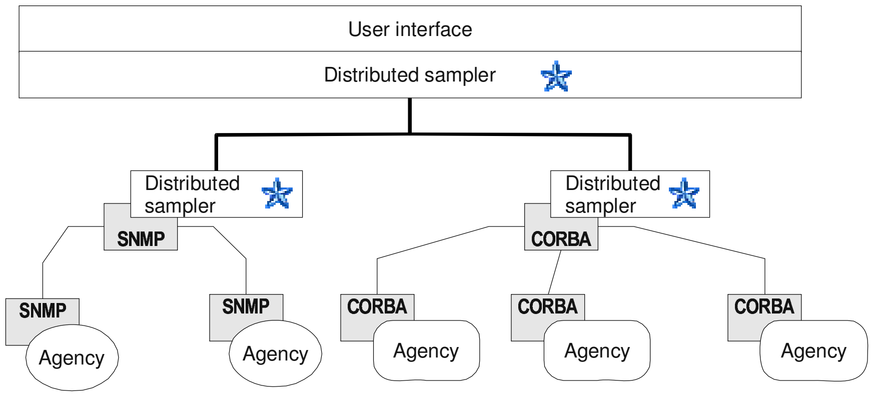


Fig. 2. Distributed sampling mechanism

The hierarchy of distributed samplers defines several "islands" of independent sampling, thus reducing upper level network load and making possible the distribution of processing load among them.

The distributed sampler (DS) has the responsibility of sampling, logging and processing raw information retrieved from the agencies. The sampling operation requires it to retrieve values from the agencies by using a specific syntax and associates them with a time-stamp. The value is then logged to an internal database with user defined capacity based on time (space to store the values retrieved during two days operation) and on the number of samples (space to store one thousand samples). Finally, the information processing requires the DS to apply some kind of algorithm or mathematical expression to the logged values. One such operation is the evaluation of the difference between two consecutive samples: delta. Another feature the DS should have is the possibility of registering callbacks to be notified when some value changes (Fig. 3).

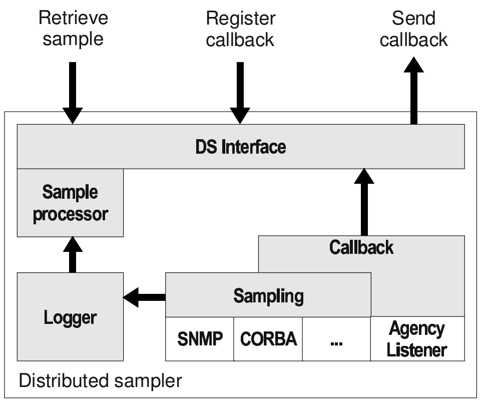


Fig. 3. Architecture of the Distributed Sampler

In other words, the DS is a mobile agent with the capability of carrying different sample processors, a logger and different sampling mechanisms. This is easily achieved by defining attributes with the type of some base class and then initialized with an instance of the desired mechanism, such as SNMP or CORBA.

3 Graphical Elements

Graphical elements are used in the communication between applications and users. In the context of mobile agents the user may interact with the agents in two scenarios:

a) for introducing and retrieving information on mobile agents and

b) for managing agent systems.

In the first variant mobile agents interact with the user and, as such, they present a user interface for receiving user input. This situation introduces problems related to the agent platform nature and its capabilities as well as with the role of the user. If the agent is running on a regular PC, it can show up graphical windows; if it is running in a WAP terminal, it can show up WML (*Wireless Markup Language*) pages. The agent should also adapt the user interface according to the user role and permissions. For example, the system will present different features to a nurse and to a doctor. In other words, user interfaces to mobile agents should change the way they look according to were they are and to whom they are speaking to [9][10].

Agent system management, on the other hand, requires a specific application to monitor and control the operation of agents and agencies that will be based on the infrastructure information system. Almost every known MA implementation provides simple tools for controlling the agent life-cycle and for creating and destroying agents as well as places. However, the management of the supporting infrastructure, namely the agencies and hosts' resources, is usually marginally considered or not considered at all. Some related work focused on these problems at the API level but disregard the user interface level [11].

Each agency is characterized by a chart showing three parameters: the CPU load, the number of mobile agents and the number of static agents. The chart shows a set of values to build an idea of their evolution. When a new value arrives, the chart is updated by dislocating the values to the left (Fig. 4).

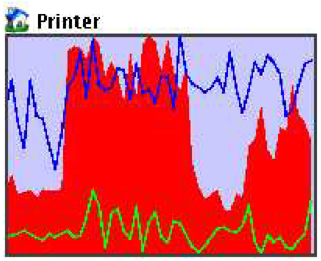


Fig. 4. Chart showing the processor load (*gray area*), the number of static agents (*light gray*) and the number of mobile agents (*dark gray*)

The main window has a search mechanism on the left which is used to locate the agencies registered in a specific region. The agency list below allows the user to position them by drag and drop on the map on the right. Then he can establish the sampling parameters for each one of them and start collecting values from the agency. A multi tabbed panel shows several layers which can represent a building floors, for example.

To get the full picture of the agents' and agencies' details, it is most useful to associate their position to a geographical map. The map may serve as background for the agencies and agents thus allowing the user to position them according to their location. A more complex scenario may include the tracing of mobile agencies together with a user location service to visualize, for example, the agency installed in his PDA or mobile phone (Fig. 5).

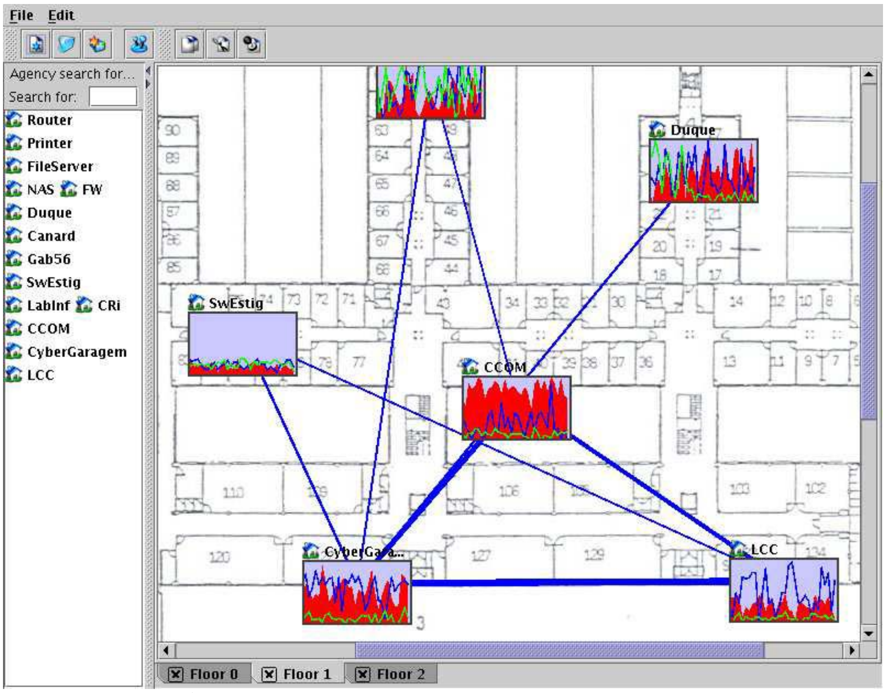


Fig. 5. User interface to the mobile agent infrastructure

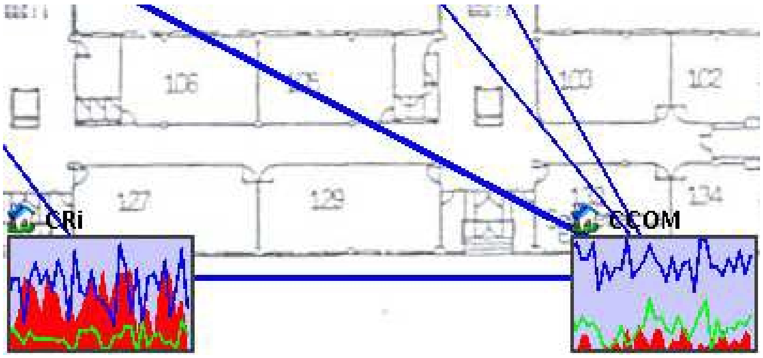
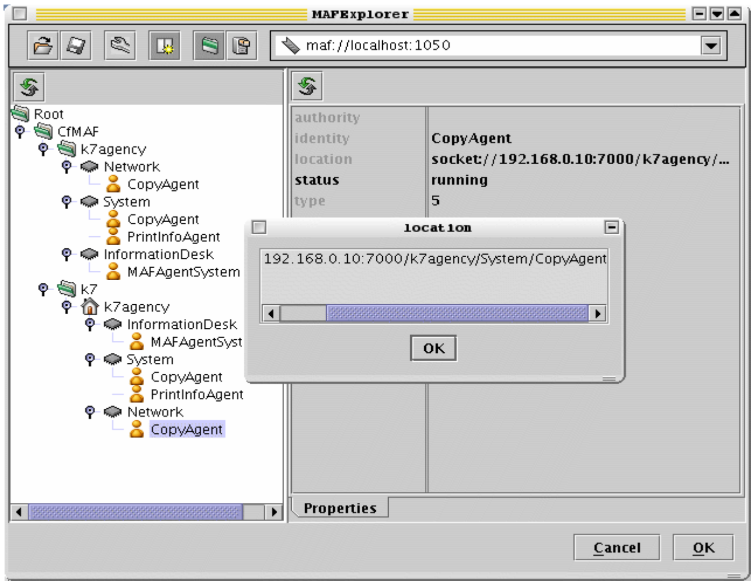


Fig. 6. Connectivity between agencies

In addition to these values, the interface also shows the connectivity between agencies. This parameter is measured as a rate of migrations between agencies, resulting from the evaluation of the number of migrations to a period of time. This value is then normalized as a percentage of all the migrations monitored and presented as lines. Different widths represent the relative percentage of migrations. The heavier the line, the heavier is the agent traffic between the connected agencies (Fig. 6).

To examine the details of agents and agencies, the user may call an explorer like tool [6]. The MAF Explorer follows the file manager paradigm with a tree view on the left and the content panel on the right. On the right side, it shows the details of the place or the agent, as selected on the tree. The grayed labels indicate that the parameter is read only. As an example, the user cannot modify the agent location, although the agent may move autonomously. A black label indicates that the user may also change its value. The agent status may be altered to suspend, resume or terminate its running status (Fig. 7).



authority	
identity	CopyAgent
location	socket://192.168.0.10:7000/k7agency/...
status	running
type	5

location: 192.168.0.10:7000/k7agency/System/CopyAgent

Fig. 7. Agency explorer

4 Implementation Details

The system is still being implemented at the time of writing of this paper, although some of the intended functionality is already working. Mobile agents support is provided by IKV's Grasshopper agent platform [12] and we are using the Sun's mapping of CORBA included in the J2SE [13] for the instrumentation objects.

The mobile agents information system is populated by the MAF interfaces and specific IDL interfaces to provide access to host instrumentation data. It was defined an IDL interface to define the mechanism to deal with host information and to implement the distributed callbacks mechanism. Data such as CPU load is retrieved from the agency host by a CORBA server object which also accept the registration of callback objects, so we have a scenario of mobile agents/CORBA to get all the information related to the navigation management system.

The monitoring system has to register a callback by acquiring a reference to the server object and then invoke the `register(callback)` method with a reference to the callback object. From this moment on it will be able to receive notifications.

5 Conclusions

In this paper we presented the approach, design and implementation issues of a mobile agent navigation manager. We discussed several related issues such as the mobile agent information system, the sampling strategies behind it and the set of graphical tools that build an intuitive graphical user interface.

Through the interface, the user will get access to the mobile agent system operation parameters and he will be able to start new tasks, modify existing ones and access information about how the system is working. We also presented concerns with the interoperability between the navigation manager with different vendors' mobile agent infrastructures by providing an all CORBA “driver” to the host and agency information.

References

1. Pham, V., Karmouch, A., "Mobile Agents: An Overview.", IEEE Communications Magazine, 1998.
2. Ibbotson, R., Gibbard, B., Stampf, D., Throwe, T., "A Mobile-Agent Based Performance-Monitoring System at RHIC", International Conference on Computing in High Energy and Nuclear Physics, Padova, Italy 2000.
3. Tierney, B., The Distributed Monitoring Framework (DMF), http://www-didc.lbl.gov/DMF/.
4. OMG, Mobile Agent Facility Specification, ftp://ftp.omg.org/pub/docs/formal/00-01-02.pdf.
5. Lopes, R., Oliveira, J., "SNMP Management of MASIF Platforms", proc. of the IFIP/IEEE International Symposium on Integrated Management - IM'2001, Seattle 2001.
6. Lopes, R., Oliveira, J., "Multi-management Schemes for MAF Platforms", proc. of the Fourth International Workshop on Mobile Agents for Telecommunication Applications (MATA'2002), 2002.
7. Simões, P., Silva, L., Boavida, F., "Integrating SNMP into a Mobile Agents Infrastructure", proc. of the Tenth IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM'99), 1999.
8. Bohoris, C., Pavlou, G., Liotta, A., "A Hybrid Approach to Network Performance Monitoring Based on Mobile Agents and CORBA", proc. of the 4th International Workshop of Mobile Agents for Telecommunications Applications - MATA'2002, 2002.
9. Braubach, L., Pokahr, A., Moldt, D., Lamersdorf, W., "Using a Model-based Interface Construction Mechanism for Adaptable Agent User Interfaces", proc. of AAMAS Workshop 16 - Ubiquitous Agents on Embedded, Wearable, and Mobile Devices, 2002.
10. Silva, A., Silva, M., Romão, A., "User Interfaces with Java Mobile Agents: The AgentSpace Case Study", proc. of the First International Symposium on Agent Systems and Applications Third International Symposium on Mobile Agents, 1999.
11. Simões, P., Marques, P., Silva, L., Silva, J., Boavida, F., "Towards Manageable Mobile Agent Infrastructures", proc. of the International Conference on Networking - ICN'01, Colmar, France 2001.
12. IKV++, Grasshopper Mobile Agent platform, http://www.grasshopper.de/.
13. Sun, Java 2 Standard Edition, http://java.sun.com/j2se/.

An Evidence-Based Mobility Prediction Agent Architecture

Nancy Samaan and Ahmed Karmouch

Multimedia and Mobile Agent Research Lab.,
School of Information Technology & Engineering (SITE), University of Ottawa
161 Louis Pasteur St. Ottawa, ON, Canada K1N-6N5
{nsamaan, karmouch}@site.uottawa.ca

Abstract. One of the major challenges in wireless environments is the provision of Quality of service (QoS) guarantees that different applications demand considering the highly dynamic nature of these environments. User mobility prediction represents a key factor for providing a seamless delivery of multimedia applications over wireless networks. Most of the existing approaches for mobility prediction presume that users travel in a-priori known pattern with some regularity; an assumption that may not always hold (e.g., a tourist in a foreign city). This paper presents a novel architecture of a mobility prediction agent (MPA) that accurately performs mobility prediction using knowledge of user's preferences, goals, and spatial information without imposing any assumptions about the availability of his movements history. Using concepts of evidential reasoning of Dempster-Shafer's theory, the MPA captures the uncertainty of the user's navigation behavior by gathering pieces of evidence concerning different groups of candidate future locations. These groups are then refined to predict the user's future location when evidence accumulate using Dempster rule of combination.

1 Introduction

Wireless technologies represent a rapidly growing area of importance. This can be attributed to the fact that small portable computers are now easily affordable and are becoming quite common in everyday business and personal life. Current wireless computing paradigms aim at providing mobile end users with *Anywhere/Anytime* access to a wide range of computing services [1], with an emphasis on highly demanding multimedia applications such as video tele-conferencing and news-on-demand.

Our goal is to build a system to support demands of multimedia applications on wireless networks. A major acceptance criterion for these applications is their high QoS requirements. Our overall solution to this problem is a multi-agent framework of intelligent agents located at different network elements. The framework is composed of three layers of agents that cooperate to dynamically adapt network behavior to maintain the delivered QoS level. At the first layer, mobility prediction agents utilize knowledge about the user to predict his future location. The predicted location is then reported to a set of QoS adaptation agents, located at the second layer, to dynamically adapt the network policies to meet the user's requirements. In the third layer, resides a set of monitoring agents that are dedicated to provide a feedback mechanism based on QoS measurements.

This paper presents a new architecture of mobility prediction agents (MPA) that accurately predict future locations of mobile users.

An ideal scenario is that the system obtains a-priori knowledge of the exact trajectory of every user. Mobility information can then be used to assist user's mobility management, network resources reservation and call admission control. While there have been several attempts to address the issue of mobility prediction, most of the existing techniques are based on the use of historical movement patterns pertinent to the users to calculate their possible future locations. These techniques are based on the assumption that the user's movements exhibit some regularity. In this case, a training phase is first required during which regular movement patterns are detected and stored. User's movement behavior may be highly uncertain and assumptions about user's movement patterns should be made with utmost care. Thus, an obvious disadvantage of most of the existing approaches is that they fail significantly whenever the user is situated in new locations, or when there is a slight change in his mobility patterns. Example scenarios are predicting the location of a tourist touring in a city, or a student navigating for the first time in university campus.

We propose a novel architecture for an MPA to accurately predict user's mobility trajectory, while overcoming the aforementioned drawbacks. The robustness of the proposed work is the result of a two-fold contribution. The first incorporates existing crucial information such as real-world maps and user's profile and preferences. The second contribution is the utilization of the rich mathematical theory of evidence as a tool for reasoning for investigating the user's behavior concerning his decisions about his future location. The idea behind such combination of knowledge and reasoning is to improve the overall competence of the prediction algorithm to be able to handle novelties taking into consideration possible uncertainty in the available information. A spatial conceptual map (SCM) along with user's available information, such as his profile, schedule, and preferences are used to extract important pieces of evidence concerning groups of possible future locations. These groups are then refined to predict the user's future location when evidence accumulates using Dempster rule of combination.

The rest of this paper is organized as follows. Section 2 presents a general description of the overall multi-agent system. In section 3 related work and existing approaches for mobility prediction are briefly discussed. Section 4 introduces the necessary background of Dempster-Shafer's theory of Evidence. The proposed mobility prediction agent architecture is described in section 5. Finally, section 6 concludes the paper.

2 Proposed Framework

The proposed architecture is a multi-agent system in which intelligent agents cooperate to predict future changes in the delivered QoS and adapt the network behavior according to these changes. As shown in Figure 1 the architecture is composed of three layers of agents. In the first layer, mobility prediction agents (MPA) utilize user's information, such as his profile, preferences and goals, to predict her possible future location. MPAs then report to a set of QoS adaptation agents (QAA), residing in the second layer, which dynamically modify network policies to best utilize network resources in order to meet the user's needs. A feedback mechanism is achieved through information obtained by monitoring agents (MA), existing on the third layer, to ensure that the applied policies perform as intended. The main focus of this paper is the first layer where MPAs are responsible of accurately predicting future locations of users and deliver this information to the QoS adaptation agents located in the second layer. Details concerning the second and third layers have been discussed in a previous work [2].

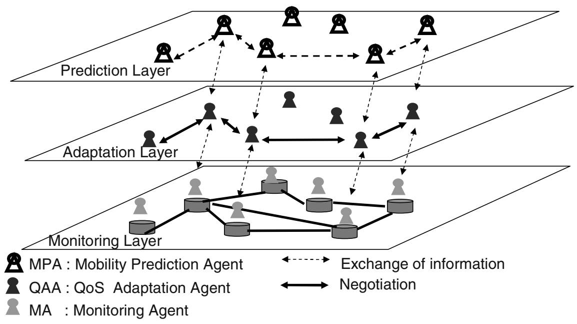


Fig. 1. Schematic description of the proposed architecture.

3 Related Work and Motivation

A number of schemes for movement prediction have been reported in literature. In [3] the user's location is determined based on her quasi-deterministic mobility behavior represented as a set of patterns stored in a user's profile. In [4] Liu et al. further pursued this method by modelling movements as repetitions of some elementary patterns and used a matching/recognition-based mobile motion prediction algorithm (MMP) to estimate her location. The main drawback of these algorithms is their sensitivity to any changes in the pattern of movement. Simulation results for the MMP algorithm, as reported in [5], show that the accuracy of the MMP decreases linearly with the increase in the randomness factor. Prediction accuracy can reach 70% for users with 30% randomness in their movements, however, it drops to almost 45% if the randomness factor increases to 50%.

To increase the accuracy of location prediction, recent research trends predict user's movement through modelling his movement by storing all movement patterns derived from his long-term history. The time-varying location probability is estimated and then used to predict the new location and the arriving time of the user. Existing approaches differ in the method by which the movement history is collected, and the way this information is analyzed. In Doppelganger [6], a system developed in MIT Labs, user's movements are collected through active badges, Unix logins, and schedule files. In [7], Bhattacharya et al. describe a tracking system for call delivery. A dictionary of the user's path updates is used as an input to a Markov model. As users move between cells the model is updated and the network tries fewer cells to successfully deliver a call. A closely related work, carried out in Georgia Institute of Technology [8], uses a GPS-based system to collect location information. The system clusters GPS data taken into meaningful locations at multiple scales. These locations are incorporated into a Markov model to predict the user's future location. In ComMotion [9], the location model is constructed from a set of learnt destinations that the user has categorized. Different pattern recognition models including Markov models and Bayes models are used for route prediction. The model presented has certain limitations. For example, the system takes time to learn locations which are not often frequented. In [10] a prediction-based location management using a multi-layer neural network (MNN) was proposed. MNNs are trained with data obtained from the user's movements for a period of time, before the prediction scheme is used.

In general, for statistical prediction methods (e.g. Markov predictors) to succeed, history patterns must exist a-priori, i.e., sufficient data must be generated to provide reliable estimates of transition probabilities. Furthermore, these models are built upon a stability assumption that the user follows the same behavior for a long period of time; an assumption that might not be always true. For example, a student might have a model built using the locations of his classes for an entire semester. When the next semester starts he may have an entirely different schedule; in this case it might take the entire semester for the model to be updated to correctly reflect the new information. A simulation study was performed in [11] using data collected within ORL (Olivetti and Oracle Research Laboratory) to track the movement of its staff for 15 days using various prediction schemes. Roughly 60% of the data was used to establish a history database and 40% of the data was used to carry out movement estimations. Although this case study involved regular movement patterns for employees in their usual workplace, the results showed that most schemes have a prediction accuracy ratio in the range 50 to 70%. Higher levels of randomness in the users' movement further degrades the prediction accuracy of these schemes.

We argue that there are two limitations for schemes relying on users' mobility patterns. The first concerns the change of his behavior. As he visits new places for which no past history is available, prediction schemes fail to work. The second is related to logging the mobility history for a long time. This causes recent changes of the user's behavior to have insignificant impact on the prediction scheme which must go through a learning phase again before it can accurately function. In [12] Soh et al. propose to overcome these limitations by assuming that a user's movement tends to follow patterns of other people in nearby places moving in the same direction. Statistics about movement patterns of other users already passed through the same cell are used to predict the user's future location.

Although it is generally accepted that learning is a necessary step in the prediction process, it is argued that it is more beneficial to learn the user's habits and behavioral patterns rather than his movements history. Ignoring this fact, other approaches fail to take into account certain observed aspects such as trip purpose and attraction zones. Other behavioral information can be used to enhance prediction schemes in order to accurately represent and deal with incompletely specified situations characterized by partial or even complete absence of knowledge about the users previous movement patterns in the current location. To emphasize the importance of developing a prediction scheme that does not impose any assumptions about the existence of a user's movement history, we consider the following three scenarios [13]. The first concerns a tourist visiting a foreign town. The second concerns a new student in campus. The final one concerns a businessman on a business trip away from home. It is clear that in these scenarios no movement history can aid in the prediction process and thus other alternative information must be used.

4 Dempster-Shafer Theory of Evidence

Dempster-Shafer theory [14, 15] has attracted considerable attention as a successful approach in dealing with problems of combining different bodies of evidence to reach decisions in situations characterized by high degree of uncertainty. The main advantage of the underlying theory of evidence over other approaches is its ability to model the narrowing of a hypothesis with the accumulation of evidence, and to explicitly represent uncertainty in the form of ignorance or reservation of judgment. It starts by assuming a Universe of Discourse Θ, also called *Frame of Discernment*, which is a set of mutually exclusive and exhaustive propositions about a domain. Let 2^Θ denote the set of all subsets of Θ. Elements of 2^Θ are the propositions in which the theory is concerned.

A function $m{:}2^{\Theta} \rightarrow [0,1]$ is a basic probability assignment bpa if it satisfies that for the empty set, m is 0; and that the sum of m over all subsets of Θ is 1. That is:

$$m(\phi) = 0, \qquad \sum_{A_i \subseteq \Theta} m(A_i) = 1 \tag{1}$$

The basic probability assignment m is referred to as *mass distribution* to distinguish it from the probability distribution. Note that it applies directly to the evidence (subsets of the frame of discernment Θ), not to the elements of Θ as in traditional probability theory.

A belief function $Bel : 2^{\Theta} \rightarrow [0,1]$ is defined such that,

$$Bel(H_i) = \sum_{A_i \subseteq H_i} m(A_i) \tag{2}$$

where $Bel(H_i)$ summarizes all our reasons to believe in a hypothesis H_i.

The theory provides a means for combining beliefs from distinct sources, known as Dempster-Shafer rule of combination. Suppose m_{E_i} and m_{E_j} are two bpa 's of the same Θ from independent bodies of evidence, E_i and E_j. The combined bpa is computed as:

$$m_{E_i} \oplus m_{E_j}(C) = \frac{\sum_{X \cap Y = C} m_{E_i}(X) m_{E_j}(Y)}{1 - \sum_{X \cap Y = \phi} m_{E_i}(X) m_{E_j}(Y)}, \textit{ for all non-empty } C \tag{3}$$

where $1 - \sum_{X \cap Y = \phi} m_{E_i}(X) m_{E_j}(Y)$ is a normalization factor. This combination rule computes a measure of agreement between two bodies of evidence concerning different propositions discerned from a common frame of discernment. An important property of the rule is that it is commutative and associative. This is desirable because evidence aggregation should be independent of the order of its gathering.

5 Mobility Prediction Agent (MPA) Architecture

Figure 2 presents a schematic description of the proposed MPA located on the user's Mobile terminal (MT). It gathers the necessary information for the prediction process and analyzes this information using Dempster-Shafer's theory in order to predict the future location of the user. The prediction process is carried out in four main phases:

- ***Information gathering:*** In this phase, the MPA collects general information that describe the landscape and environment of the mobile user.
- ***Evidence Extraction:*** In this phase, concepts of the Dempster-Shafer's theory are applied to the gathered information to generate hypotheses and bodies of evidence of the potential future location. In particular, the following sets are identified:
  - The set of competing hypotheses and discernment frame of future locations.
  - The set of evidence which are deemed relevant to the prediction process and act as the input to the prediction framework.
  - A consistent method of estimating and assigning belief mass to the hypotheses given a piece of evidence.

  By matching characteristics of candidate locations with user preferences, interests, and goals, the MPA extracts different pieces of evidence that either support or undermine the assertions regarding the future location of the user.

– ***Decision making:*** Once different pieces of evidence are collected, The MPA combines them to yield a final mobility prediction decision.
– ***Path finding:*** The final phase in the mobility prediction process is to determine the exact user's trajectory based on the predicted future location.

As shown in Figure 2 the MPA is composed of two main modules: a *reasoning engine* which is responsible for the first three stages of the prediction process and a *path finder* module that performs the final stage of finding the user's expected trajectory. The following subsections present further details of the prediction process described above.

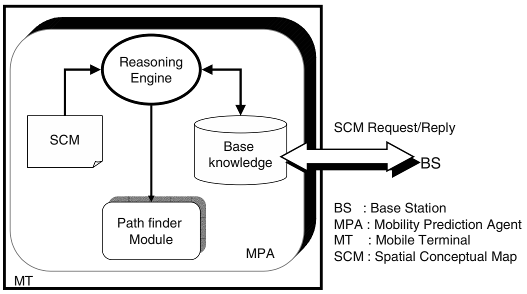


Fig. 2. Mobility Prediction Agent (MPA) Architecture

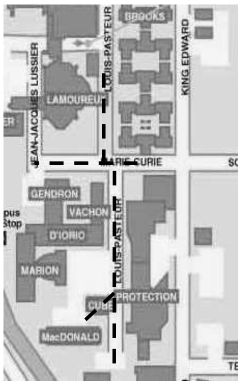
5.1 Information Gathering

Information regarding the landscape and environment of the user is represented using a *Spatial Conceptual Map* (SCM) [16]. As defined in [16] an SCM is "an abstraction of a real map representing a portion of the urban environment". An SCM contains representations of a set of *landmark objects* $O_{\text{SCM}} = \{O_1, O_2, \cdots, O_n\}$ and a set of *way areas* $W_{\text{SCM}} = \{W_1, W_2, \cdots, W_m\}$. Way areas define areas on which users can move ,e.g., streets, roads, highways, or simply trajectories and virtual connections between objects. Landmark objects are places of interests for users such as buildings and monuments. A way area W_x is further partitioned into a set of k consecutive segments called *Way Elementary Areas* (WEA), such that $W_x = \{\alpha_1^x, \alpha_2^x, \cdots, \alpha_k^x\}$. Figure 3 shows a portion of Ottawa university campus map (Figure 3(a)) and its SCM representation (Figure 3(b)).

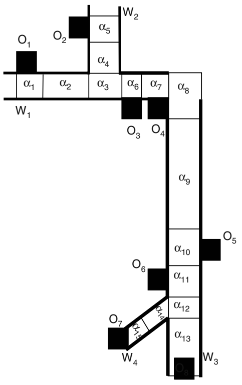
A characterization function C_O is associated with each landmark object O_i, where $C_O(O_i)$ represents basic information about O_i. For example, if O_i is a restaurant, then $C_O(O_i) = \{c_1 = \text{Food}, c_2 = \text{Chinese}, c_3 = \text{moderate prices}\}$. For a museum O_j, $C_O(O_j) = \{c_1 = \text{Historic}, c_2 = \text{Greek}, c_3 = \text{no cost}\}$. In general, $C_O(O_i)$ can be represented by n characteristics of a landmark object O_i, such that $C_O(O_i) = \{c_1, c_2, \cdots, c_n\}$.

An SCM is represented by a sparse matrix, *the matrix of orientation, adjacency, and characterizations* (MOAC). The columns and rows of an MOAC represent the WEAs of the SCM. Each cell (i, j) of the MOAC, where i and j correspond to the column α_i and row α_j respectively, contains: information about the WEA α_i orientation to another WEA α_j, if $i \neq j$, and adjacency to landmarks along with their characteristics, if $i = j$. When α_i and α_j are adjacent, then MOAC(i,j) contains the orientation of possible displacement between the two WEAs. If a landmark O_x is close to α_i then $\text{MOAC}(i,i) = (O_x, C_O(O_x))$. Figure 3(c) represents the MOAC of Figure 3(b).

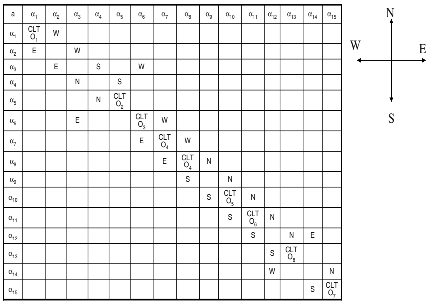
It is assumed here that the SCM information is maintained by a spatial agent (SA) located at each base station (BS) and can be obtained through MPA requests to the BS.



(a) Campus map



(b) SCM map



a	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}	α_{11}	α_{12}	α_{13}	α_{14}	α_{15}
α_1	CLT O_1	W													
α_2	E		W												
α_3		E		S		W									
α_4			N		S										
α_5				N	CLT O_2										
α_6			E			CLT O_3	W								
α_7						E	CLT O_4	W							
α_8							E	CLT O_4	N						
α_9								S		N					
α_{10}									S	CLT O_5	N				
α_{11}										S	CLT O_6	N			
α_{12}											S		N	E	
α_{13}												S	CLT O_8		
α_{14}												W			N
α_{15}														S	CLT O_7

(c) MOAC representation

Fig. 3. An Example of an SCM and MOAC representations of a spatial map

5.2 Evidence Extraction

The goal of this step is to apply concepts of Dempster-Shafer's theory to the information gathered in the previous step along with the information available concerning the user to generate hypotheses and bodies of evidence that will form the input to the next phase. The functionalities of this phase is carried out by the reasoning engine of the MPA. The frame of discernment Θ is generated from all potential future destinations. i.e., $\Theta = \{O_i : O_i \in O_{\text{SCM}}\}$, where O_{SCM} represents the target locations obtained from the SCM map.

Next, the reasoning engine uses the domain 2^{Θ}, the set of all subsets of Θ, to generate different hypotheses and applies different rules to assign a belief mass value to each hypothesis. This process is performed based on different premises that represent different bodies of evidence that can foretell the user's future behavior. To be precise, the following types of premises are used by the reasoning engine to structure the frame of discernment into groups of candidate future locations represented by different hypotheses.

- ***User's interest in nearby places:*** The MPA uses its knowledge base to store the user's preferences for particular domains. Furthermore, rules for matching interests are also loaded in the agent's knowledge base to aid the reasoning engine in determining the user 's interest in certain locations. A hypothesis H_{c_i} is constructed through grouping locations whose characteristics fall into one domain of user's interests c_i, i.e.,

$$H_{c_i} = \{O_i : O_i \in \Theta, c_i \in C_O(O_j)\} \tag{4}$$

  A belief mass value ω_i is associated with H_{c_i}, such that:

$$m(H_{c_i}) = \omega_i \tag{5}$$

  where ω_i represents the degree of the user's interest in domain c_i.
- ***User's schedule constraints:*** User's schedule constraints represent another strong body of evidence that can help in predicting the user's future location. If O_s represents the location of the earliest scheduled appointment and t_s the time left before the scheduled appointment takes place, then a hypothesis H_{t_i} can be defined as:

$$H_{t_i} = \{O_i : O_i \in \Theta, (t(O_c, O_i) + t(O_i, O_s)) < t_i\} \tag{6}$$

  where O_c is the user's current location and $t(O_i, O_j)$ is the time taken to visit O_j starting from O_i. A belief mass value P_{t_i} is associated with H_{t_i} such that:

$$m(H_{t_i}) = P_{t_i} = \text{Probability}(\text{user will visit } H_{t_i}/(O_c, O_s) = t_s) \tag{7}$$

- ***User's goals and tasks:*** Another set of hypotheses, H_{g_i}, concerning candidate locations grouped according to their relation to one of the user's goals, can be constructed. If a user's goal g_i is associated with certain characteristics $C(g_i) = \{c_1^{g_i}, c_2^{g_i}, \cdots, c_n^{g_i}\}$, a hypothesis H_{g_i} can be generated such that

$$H_{g_i} = \{O_i : O_i \in \Theta, C(g_i) \subseteq C(O_i)\} \tag{8}$$

  A belief mass value i_{g_i} is associated with H_{g_i} such that:

$$m(H_{g_i}) = i_{g_i} \tag{9}$$

  where i_{g_i} represents the importance of the user's task g_i.

It is worth mentioning here that since the Dempster-Shafer theory supports reasoning with incremental evidence, the evidence extraction phase can be further customized to include or discard different bodies of evidence, other than the ones mentioned above, depending on the amount of available information about the user's preferences.

5.3 Decision Making

To determine the user's future predicted location, the reasoning engine combines each pair of hypothesis-belief mass using the Dempster rule of combination of Equation (3) to come up with a conclusion about the user's most likely future location.

Since the Dempster rule of combination is commutative and associative, these available bodies of evidence can be combined in any order. The result is a list of candidate locations together with their corresponding belief values which describes the degree of support for each candidate location. The location with the highest belief value is the predicted future location. Figure 4 shows an example of the prediction process for a student in campus. Applying Equation (3) on m_{E_1}, m_{E_2}, m_{E_3}, and m_{E_4} yields the combined belief distribution in Figure 4(b). The results show the belief in each of the candidate hypotheses. The predicted future location is the computer science lab. represented by O_3.

5.4 Path Finding

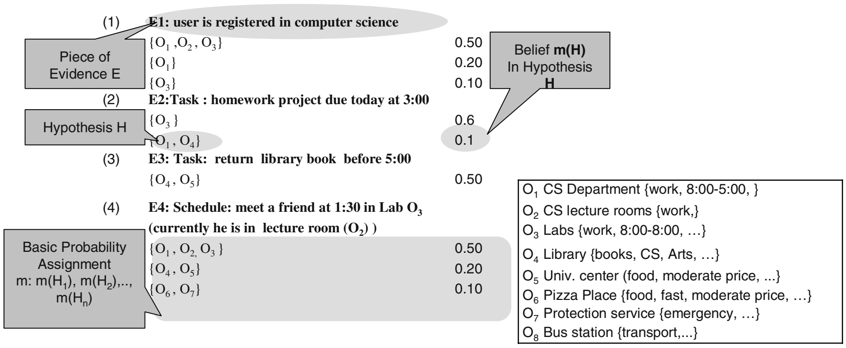
Once the future location is predicted, the path finder module uses the orientations of the MOAC to determine the path from the current location represented by the WEA α_{current} to the predicted future location $\alpha_{\text{predicted}}$. The outcome is a list of couples $(\alpha_x, orientation)$. Consider the SCM of Figure 3(b), where $\alpha_{\text{current}} = \alpha_5$ and $\alpha_{\text{predicted}} = \alpha_6$, then the user's trajectory can be represented by: $((\alpha_5, \text{S}),(\alpha_4, \text{S}),(\alpha_3, \text{E}), (\alpha_6, *))$, where $*$ is the target location. If more than one path from α_{current} to $\alpha_{\text{predicted}}$ exists, then different strategies such as minimizing the distance can be used based on the user's common behavior.

6 Conclusions and Future Work

In this paper we have presented a novel framework for an MPA based on Dempster-Shafer's theory of evidence. Uncertainty of the user's navigation behavior was captured by gathering pieces of evidence concerning different groups of candidate locations. These groups were then refined to predict her future location when evidence accumulate using Dempster rule of combination. In contrast to other approaches no assumptions are imposed concerning the availability of the user's history. Furthermore, since the theory supports reasoning with incremental evidence, evidence other than the ones discussed can be included at a later stage. Currently we are evaluating our work through implementation. In future work, we plan to investigate the role of our approach in enhancing different network tasks such as handoff management.

References

1. L. Kleinrock, "Nomadicity: Anytime, Anywhere In A Disconnected World", *Mobile Networks and Applications*, vol. 1, n. 4, pp. 351–357, Jan. 1996.
2. N. Samaan and A. Karmouch, "Prediction-Based Policy Adaptation for QoS Management in Wireless Networks", in *POLICY 2003, Italy*, Jun. 2003.
3. S. Tabbane, "An alternative strategy for location tracking", *IEEE Journal on Selected Areas in Communications*, vol. 13, Jun. 1995.
4. G. Liu and G. Maguire, "A Class of Mobile Motion Prediction Algorithms for Wireless Mobile Computing and Communication", *ACM International Journal on Wireless Networks*, 1996.


(1) E1: user is registered in computer science
$\{O_1, O_2, O_3\}$ 0.50
$\{O_1\}$ 0.20
$\{O_3\}$ 0.10
Piece of Evidence E
Belief m(H) In Hypothesis H
(2) E2:Task : homework project due today at 3:00
$\{O_3\}$ 0.6
$\{O_1, O_4\}$ 0.1
Hypothesis H
(3) E3: Task: return library book before 5:00
$\{O_4, O_5\}$ 0.50
(4) E4: Schedule: meet a friend at 1:30 in Lab O_3 (currently he is in lecture room (O_2))
$\{O_1, O_2, O_3\}$ 0.50
$\{O_4, O_5\}$ 0.20
$\{O_6, O_7\}$ 0.10
Basic Probability Assignment m: $m(H_1), m(H_2),.., m(H_n)$
O_1 CS Department {work, 8:00-5:00, }
O_2 CS lecture rooms {work,}
O_3 Labs {work, 8:00-8:00, …}
O_4 Library {books, CS, Arts, …}
O_5 Univ. center (food, moderate price, ...}
O_6 Pizza Place {food, fast, moderate price, …}
O_7 Protection service {emergency, …}
O_8 Bus station {transport,...}

(a) Sample Prediction

#	Hypothesis	Belief
1	$\{O_1\}$	0.1523
2	$\{O_3\}$	0.538
3	$\{O_1, O_2, O_3\}$	0.204
4	$\{O_4\}$	0.018
5	$\{O_1, O_4\}$	0.006
6	$\{O_4, O_5\}$	0.052
7	$\{O_6, O_7\}$	0.09
8	$\{O_1, O_2, \ldots, O_8\}$	0.018

(b) Combination of Evidence

Fig. 4. An example of evidence-based mobility prediction

5. Y. Liu, G. and Q. Maguire, G., "A predictive mobility management algorithm for wireless mobile computing and communications", in *4th IEEE Int. Conf. on Universal Personal Communications. 1995*, pp. 268–272, Nov. 1995.
6. J. Orwant, "Doppelganger goes to school: Machine learning for user modeling", in *Master's thesis, MIT Media Laboratory*, Sept. 1993.
7. A. Bhattacharya and K. Das., S., "Lezi-update: An information-theoretic approach to track mobile users in PCS networks", *In Mobile Computing and Networking,*, pp. 1–12, 1999.
8. D. Ashbrook and T. Starner, "Learning Significant Locations and Predicting User Movement with GPS.", in *Proc. of ISWC02, Seattle, WA.*, Oct. 2002.
9. Chris Schmandt Natalia Marmasse, "A User-Centered Location Model", *Personal and Ubiquitous Computing*, vol. 6, n. 5-6, pp. 318 – 321, Dec. 2002.
10. V. Kumar and P. Venkataram, "A Prediction based Location Management using Multi-Layer Neural Networks", *Jl. of IISc*, vol. 82, n. 1, pp. 7–21, 2002.
11. J. Chan, S. Zhou and A. Seneviratne, "A Qos Adaptive Mobility Prediction Scheme for Wireless Networks", in *IEEE GLOBECOM 98, Nov. 1998*, Oct. 1998.
12. S. Soh, W. and S. Kim, H., "QoS Provisioning in Cellular Networks Based on Mobility Prediction Techniques", *IEEE Communications Magazine*, pp. 86–92, Jan. 2003.
13. B. Schmidt-Belz, M. Makelainen, A. Nick1 and S. Poslad, "Intelligent brokering of tourism services for mobile users", in *ENTER2002*, 2002.
14. P. Dempster, A., "A Generalization of Bayseian Inference", *Journal of Royal Statistics Society Series,30,*, pp. pp.205–247, 1968.
15. G. Shafer, *A Mathematical theory of evidence*, ch. 3, Princeton Universal Press, 1975.
16. D. Kettani and B. Moulin, "A Spatial Model Based on the Notions of Spatial Conceptual Map and of Object's Influence Areas", in *COSIT 1999, Stade, Germany*, pp. 401–416, Aug. 1999.

A Novel Architectural Framework for Adapting Mobile Agents to Host Variation

Roch H. Glitho[1], Sylvain Methot[2], and Samuel Pierre[3]

[1] Ericsson Research / Concordia University, Roch.Glitho@ieee.org
[2] Sylvain.Methot@polymtl.ca
[3] Ecole Polytechnique, Montreal, Canada, Samuel.Pierre@polymtl.ca

Abstract. Mobile agents have emerged in the mid-90s. They are programs that can start execution in a host, suspend execution, then move to another host to resume execution. The current Internet infrastructure comprises an extensive range of hosts. Clients range from high-end personal computers (PC) with a lot of memory capacity / processing power, to memory capacity / processing power constrained personal digital assistant (PDA). Mobile agents that roam the Internet need to adapt to these constraints. This paper proposes a novel architectural framework for adapting mobile agents to host variation. Its focus is on adaptation to memory constraints. The framework consists of dynamically partitioning the agent when memory is scarce, and re-assembling it when memory is abundant. We have applied it to the mobile agents of a service architecture for Internet Telephony. The results including the prototype and the measurements are presented. They indicate that the overhead due to the partitioning / re-assembling is low. This makes the framework very appealing, especially as it can be applied to any mobile agent, provided that the set of basic assumptions we have made is valid.

1 Introduction

Mobile agents have emerged in the mid-90s. They are programs that can start execution in a host, suspend execution, then move to another host to resume execution [1,2,3]. The current Internet infrastructure comprises an extensive range of hosts. Clients range from high-end personal computers (PC) with a lot of memory capacity/processing power, to memory capacity/processing power constrained personal digital assistant (PDA). Mobile agent platforms are now available for most of these hosts including the memory capacity/processing power constrained PDA.

This paper proposes an architectural framework for adapting mobile agents to host variation. The focus is on adaptation to memory capacity, and the feasibility is demonstrated using a mobile agent based -service architecture for Internet Telephony. The next section derives the requirements and examines related work. The framework is introduced after that. The fourth section is devoted to the feasibility. It presents the prototype and the measurements. In the conclusions, we provide a summary, present the lessons we have learned, and identify the items for future work.

2 Assumptions, Requirements, and Related Work

The framework is based on partitioning. The assumptions, requirements and related work are presented in this section.

2.1 Assumptions

They are as follows:

1. The mobile agent is designed with partitioning in mind. This assumption has three implications. The first is that the agent is designed as a set of loosely coupled partitions that can be executed either at the host where the agent resides, or at a remote host. The second is that the agent is able to get information on the memory available at the target host, prior to migration. The last is that the agent can perform operations that are dedicated to its adaptation to host variation.
2. There is at least one host in the network, with enough memory capacity, to accommodate the partitions the mobile agent cannot keep locally because of memory capacity constraints.
3. Every host to which the agent can potentially migrate to, has the minimal memory capacity required by the framework. As explained later in the paper, this is due to the fact that the framework requires the agent to always keep some partitions such as the partition that selects the partitions to keep when memory is scarce. The required memory capacity is also estimated later in the paper.

2.2 Requirements

They are as follows: **1. No impact on service behaviour**
Users should not perceive any major difference in the service(s) provided by the agent even when the agent is partitioned. The service should behave the same way, especially in terms of performance. **2. Transparency**
The process should be completely transparent to the end-user. His intervention should not be required. **3. Easy deployment**
The architectural framework should be as simple as possible. It should not put additional requirements (e.g. memory) on the hosts the agent visits. **4. Minimal service interruption**
The time during which the user does not have access to the service provided by the agent (if any) should be minimal. **5. Minimal completion time**
The process should take as little time as possible. **6. Dynamic partitioning**
The partitioning should be dynamic because the number of partitions the agents keep should not be fixed. Furthermore it is necessary that the agent is able to send away (or claim back) partitions even when it is running, especially as some agents run continuously.

2.3 Related Work

Several approaches have been proposed over the years for adapting applications to host variations. Examples include process migration and mobile agent based application partitioning. In process migration, applications aim at harnessing the processing power of workstations that are idle on a network. These applications are partitioned and adapt to the variations in the busy/ idle state of the workstations. Partitions are sent away for remote execution on idle workstations and sent back to the initial hosts, when the workstations are claimed back by the owners. Mosix [4] and Sprite [5] are classical examples.

Process migration is not suited to the adaptation of mobile agents to host variation for a host of reasons. It usually requires changes to operating systems, while most mobile agents nowadays run on top of the Java Virtual Machine and are operating systems independent. Furthermore, it assumes that the host has enough memory / processing power to accommodate the application in its entirety, because partitions residing on foreign hosts can always come back home, when they are evicted. Such an assumption cannot be made for mobile agent adaptation to host variation because the very problem is that they hosts do not have enough memory to accommodate the agents in their entirety.

A mobile agent based application partitioning scheme has been proposed by O. Koskimies and K. Raatikainen [6]. Applications are partitioned as mobile agents and the partition is fully dynamic in the sense that the component agents can move anytime while the application is running. This mobility is required in order to be able to adapt to network conditions because these conditions can change anytime, especially in wireless environments. The scheme is pertinent for the problem at hand. However, it has a serious performance problem because when memory / processing power becomes abundant, the partitions continue using inter-agent communications mechanisms although they reside on the same host. The penalty in terms of performance is quite heavy, as shown later in the paper by the measurements we have made.

3 The Framework

Figure 1 depicts the framework. It is made of the partition storage component, the partition handling operation, the partition dispatching sub-operation, and the partition selection sub-operation.

The partition storage agent (PSA) stores the partitions the agent being partitioned elects to drop because of memory constraints on the host to visit next. It is either fixed or mobile and resides on a host distinct from the host on which the agent being partitioned resides. As per our assumptions, this host has the required memory capacity to accommodate all the partitions the agent elects to drop. The partition handler is a high level operation the agent performs for adaptation purpose. It relies on two sub-operations: partition dispatching and partition selection. Partition dispatching sends away the partitions the agent has elected to drop when memory is scarce, and gets back the partitions the agent has decided to claim back when memory becomes abundant. Partition selection selects the partitions to drop/claim back. We illustrate below how the scheme works in terms of what the agent does before moving to a new host (pre-migration scheme), what it does as soon it reaches a new site (post migration scheme) and also in terms of how partitions are selected.

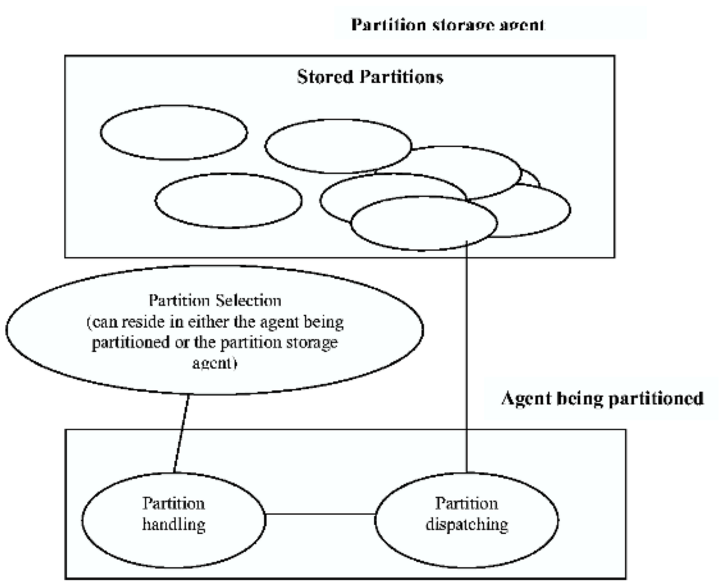


Fig. 1. The Framework

3.1 Pre-migration and Post-migration Schemes

We assume that the agent has N partitions. For simplicity sake, we also assume that the partitions have the same size. We have two options for selecting the M partitions the agent should have when it reaches the target host. The first consists of making sure the agent keeps as many partitions as possible from the set it currently has. This will reduce the number of partitions to send away / claim back. The second consists of asking the partition selector to select the M partitions independently of the N partitions the agent currently has.

Let us now look more closely at the first option. If M is less than N, then the partition handler will ask the partition selector to select the M partitions that should be kept. The remaining N-M partitions are sent to the PSA. If it is greater, then the partition selector is asked to select the M-N partitions the agent should download as soon as it reaches the target. If M is equal to N, then the agent will just move to the new target with the partitions it already has.

In the second option, M “new" partitions are anyhow selected from the set of all existing partitions. When M is equal to N, an interesting case is when none of the partitions currently kept by the agent is selected. The agent, prior to its migration will have to send to the PSA the N partitions it has, and download the N partitions selected by the partition selector. This will lead to 2N exchanges between the PSA and the agent being partitioned.

The first option is described below. As shown later in the paper, the performance of each option depends on the schemes used for partition selection. The following steps are taking by partition handling, prior to the departure of the agent.

1. It gets information on the memory capacity available at the target host
2. It computes the number M of partitions the target host can accommodate
3. If N = M then it moves to the host
4. If N > M then it:
– Asks partition selection to select the N-M partitions to drop
– Asks partition dispatching to send the elected partitions to the PSA
– Builds the proxies through which the dispatched partitions can be accessed remotely
– Moves to the target host
5. If N < M then it:
– Asks the partition selector to select the M-N partitions to download when it reaches the target host
– Moves to the target host

When the agent reaches the new host, partition handling:
1. Checks if there are partitions to be downloaded
2. If there are partitions to be downloaded, it:
– Asks the partition dispatching to download them
– Destroy the proxies that were used to remotely execute the partitions

3.2 Partition Selection Criteria and Scheme

Criteria. There are three main criteria. **1. Mandatory local execution vs. optional local execution**
Some partitions can be executed only locally. A good example is a partition that includes a graphical user interface. If a choice is to be made between a partition that can be executed locally, and another partition that can be executed both locally and remotely, the preference should go to the first when it comes to the choice of the partition the agent should keep. **2. Real time execution vs. non real time execution**
Some partitions need to be executed in real time, meaning there is an upper bound to the time lag between the invocation of the partition and the start of the execution. If a choice is to be made between a partition with real-time constraints and a partition with no real time constraint, the preference should go to the first when it comes to the agent should keep. **3. High execution probability vs. low execution probability**
Some partitions are highly likely to be executed in the near future while others are not. If a choice is to be made between a partition with a high execution probability and a partition with a low execution probability, the preference should go to the first when it comes to the partition the agent should keep.

Scheme. The time lag between the invocation of a partition and its execution can be quite important in our architectural framework. This happens for instance when the partition to be executed can be executed only locally, but happens to be in the PSA. The mobile agent will have to send to the PSA one of the partitions it carries, then claim back the partition to be executed. The key goal assigned to the partition selection scheme is to keep as low as possible the average time between the invocation of partitions and the beginning of the actual execution, while meeting the three criteria discussed above. In this scheme, we propose to rank the criteria and to use the ranking in the selection algorithm. This ranking will allow the selection of the proper partition when for instance there is space for only one partition on the target host, and a choice is to be made between two or three of the partitions listed below:

1. a partition which can execute only locally
2. a partition with real time constraints
3. a partition with a high execution probability

Let us assume that we need to select M partitions from a set of N partitions, with N>M and let us assume the following ranking (decreasing order of priority)

1. Partitions that can execute only locally
2. Partitions with real time constraints
3. Partitions with high execution probability

The partition selector takes the following actions.

1. It selects the partitions that can execute only locally
2. If there is space left it selects the partitions with real time constraints
3. If there is space left it selects the partitions with high execution probability

Algorithms using different criteria ranking schemes can be easily derived from the one above. The execution probability may vary during the whole life of the agent, while the two other criteria are fixed. When the ranking above is used, it makes sense for partition handling to use the first option (i.e. keep as many partitions as possible from the set of partitions the agent already has) of the pre-migration scheme. If the high execution probability vs. low execution probability was ranked the highest it will make more sense to use the second option.

Several algorithms have been proposed over the years for predicting the execution probability, especially in the context of operating systems [8]. We do not propose any new algorithm in this paper. However, as most of the algorithms rely on what has happened in the past, we propose a scheme for collecting and processing the information. The scheme includes a statistics handler that is part of the partition selector. The partition handler sends periodically execution statistics to this statistics handler.

4 Feasibility

We successively introduce the mobile agents to which we have applied the framework, the prototype and the measurements.

4.1 Mobile Service Agents for Internet Telephony

We have applied the framework to the mobile service agents (MSA) of the service architecture for Internet Telephony presented in reference [8]. The architecture aims at provisioning value added services (i.e. anything that goes beyond two party voice call) in Internet Telephony and two of the authors of this paper have contributed to its design. It is depicted below by figure 2.

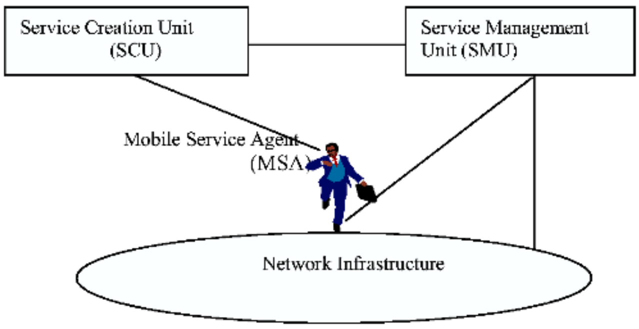

Service Creation Unit (SCU)
Service Management Unit (SMU)
Mobile Service Agent (MSA)
Network Infrastructure

Fig. 2. Mobile agent based service architecture for Internet Telephony

The MSA and the service units are briefly introduced below. The mobile service agent (MSA) is the key entity. It acts as a folder, and carries the services to which end- users have subscribed. For each of the services, it carries the logic and the data. It has a coordinator that allows it to perform a well-defined set of operations such as starting/stopping the execution of the services it carries.

There are two types of service units in the architecture, the service creation unit (SCU) and the service management unit (SMU). The SCU allows the creation of the services to which users could subscribe. The SMU manages subscriber and service life cycles. Its functionality includes the publication of the list of available services, the creation and upgrading of MSA.

It is important to note that the assumptions we have made for the framework are valid here:

1. The MSA is designed with partitioning in mind. It is made of a co-ordinator, operations and services. The operations are designed as loosely coupled partitions. The same applies to the services the MSA carries. The information on the available memory at the target host is stored in the SMU and the MSA can access it. The operations required for partitioning have been implemented as dedicated operations as shown later in the paper.
2. The SMU has enough memory capacity, to accommodate the partitions the mobile agent cannot keep/execute locally because of processing power / memory capacity constraints.
3. Every host to which the agent can potentially migrate to in our experimental setting has the minimal required memory capacity as shown later in the paper.

4.2 Prototype and Measurements

The prototype has been implemented as an extension to the prototype of the architecture described above. The PSA has been implemented as a fixed agent residing on the SMU. Partition handling has been implemented as operation. Partition selection and partition dispatching have been implemented as sub- operations. The following schemes have been implemented:

- Pre-migration
- Post-migration
- Partition selection

The statistics handler has not been implemented. Partition selection has been implemented by assigning random execution probability to the partitions. We used 3 identical 400 MHz Intel Pentium II desktops (128 Mbytes RAM) and 4 identical 400 MHz Pentium III laptops (256 Mbytes RAM). All these workstations were running Windows NT 4.0. The SMU was running on one of the 3 desktops. The two other desktops and the four laptops were used as user terminals.

Each workstation was running the Grasshopper 2.1 mobile agent platform. The workstations were connected to the same segment of a 100 Mb/second Ethernet in our local area network. The network latency was 40 ms. The measurements were performed during night in order to minimise the global network load.

We have made measurements on two of the requirements of the framework:

- No impact on service behaviour
- Easy deployment

No impact on service behaviour. A key factor that impacts service behaviour in partitioning is inter-partition communication. If it takes too long, then the user will perceive a difference in service behaviour. We have compared our framework to the mobile agent based application partitioning proposed by by O. Koskimies and K. Raatikainen [7]. We simulated mobile agent based application partitioning by implementing as mobile agent one of the operations the MSA performs.

We have measured the time that elapses between the invocation of the operation by the co-ordinator and the beginning of the execution of the operation, the co-ordinator and the operation being on the same host. Figure 3 depicts the result. Our framework outperforms mobile agent based application partitioning by a factor of at least 500.

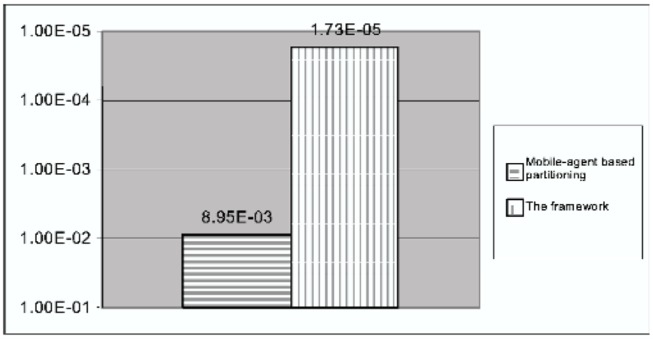


Fig. 3. Inter-Partition Communication Time

Easy deployment. Some of the partitions are mandatory and the MSA needs to always keep them. An example is the partition handling. Another example is the partition dispatching. A key factor that will influence the deployment of the framework is the minimal amount of memory the target host must have.

We have measured the amount of required memory and have compared it to the amount required by a mobile agent with no logic. The amount of memory required for instantiating the two agents, were measured, and the amount of memory required after the instantiation was also measured.

Figure 4 and figure 5 depict the results. Around 13 kbytes are needed in addition to what is needed to create a mobile agent with no logic (around 68 kbytes), using Grasshopper. After instantiation, it is around 912 kbytes in addition to the 3568 kbytes required by an agent with no logic.

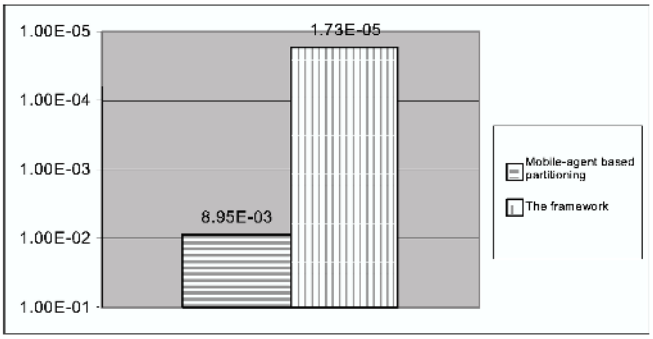


Fig. 4. Memory Required for Instantiation

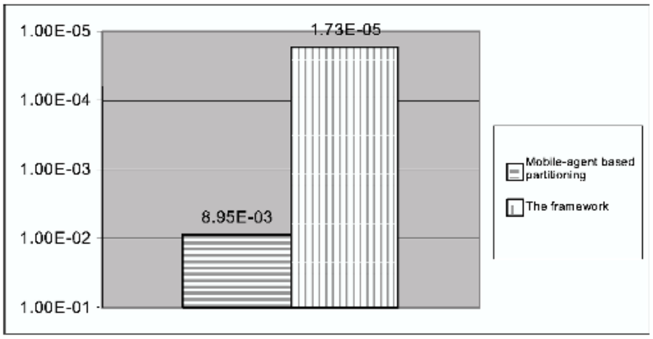


Fig. 5. Memory required after instantiation

5 Summary, Evaluation, Lessons Learned, and Items for Further Work

This paper has proposed a novel framework for adapting mobile agents to host variations. The framework is made of a partition storage agent, a partition handling operation, a partition selector sub-operation and a partition dispatching sub-operation. The partition selector is an agent which resides on a host with enough memory capacity / processing power to accommodate the partitions the agent elects to drop. Partition handling co-ordinates the partitioning process from the agent perspective. It relies on partition selection and on partition dispatching. Partition selection selects the partitions to drop. Partition dispatching sends the partitions to the partition storage agent when memory is scarce and gets them back when memory becomes abundant.

We have applied the framework to the mobile service agents of an architecture to which two of the authors of this paper have contributed. The measurements made show that the quantifiable requirements are met. The impact on service behaviour cannot be detected by the end-user because the process is much faster than the mobile agent based application partitioned proposed in the state-of-the-art. The additional memory required by the scheme is insignificant. This makes the framework easy to deploy in memory constrained environment. The overhead due to the partitioning and the re-assembly is low and has a minor impact on the completion time. The non-quantifiable requirements such as transparency and dynamic partitioning are met by design.

A first lesson we have learned is that performance can be significantly improved in mobile agent based - application partitioning if the agents that make the application are re-assembled when they are on the same host. Our framework targets the partitioning of mobile agents. However, it can be easily adapted to the partitioning of applications using mobile agents. Another lesson we have learned is that the most difficult issue in adapting mobile agents to host variation is the partition selection scheme. Although the partition selection scheme proposed in this paper relies on the ranking of criteria, we have not yet tackled the execution probability issue. In future work we will look at the schemes that will allow the computation of this probability using the history of execution.

The framework as currently defined relies on the assumption that there is a host in the network on which all the partitions the agent cannot handle are off-loaded. In future work we will loosen the assumption by considered a set of hosts, each host having the possibility of accommodating some of the partitions, but not all. We will also consider the possibility of partitions being evicted from some hosts and having to be relocated.

References

1. D. Chess et al., Mobile Agents: Are They a Good Idea?, IBM Research Report, RC 19887 (88465), 1994
2. A. Karmouch and V. A. Pham, Mobile Software Agents: An Overview, IEEE Communications Magazine, July 1998, Vol.36, No7
3. M.K. Perdikeas et al., Mobile Agents Standards and Available Platforms, Computer Networks, Vol.31, No19, August 1999, pp. 1999–2016.
4. A. Barak and R. Wheeler, MOSIX: An Integrated Multiprocessor Unix, in D. Milojicic et al. (eds), Mobility, Processes and Agents, ACM Press, Addison Wesley, Reading, 1998, pp. 42–53

5. F. Douglis and J. Ousterhout, Transparent Process Migration: Design Alternatives and the Sprite Implementation, in D. Milojicic et al. (eds), Mobility, Processes and Agents, ACM Press, Addison Wesley, Reading, 1998, pp. 56–86
6. Oskari Koskimies, K. Raatikainen, Partitioning Applications with Agents, Second International Workshop on Mobile Agents for Telecommunication Applications (MATA 2000), pp. 79–93
7. A. Tanenbaum and A. Woodhull, Operating Systems: Design and Implementation, Prentice Hall 1999
8. B. Emako, R.H. Glitho and S. Pierre, A Mobile Agent based Advanced Service Architecture for Wireless Internet Telephony: Design, Implementation and Evaluation, IEEE Transactions on Computers, Vol. 52, No. 6, June 2003

Combining Immune Systems and Social Insect Metaphors: A Paradigm for Distributed Intrusion Detection and Response System

Noria Foukia: noria.foukia@cui.unige.ch
Salima Hassas: hassas@bat710.univ-lyon1.fr
Serge Fenet: sfenet@bat710.univ-lyon1.fr
Paul Albuquerque: albuquer@eig.unige.ch

CUI - University of Geneva
24 rue du Général Dufour
CH-1211 Geneva 4, Switzerland
LII - Ecole d'ingénieurs de Genève, HES-SO
CH-1202 Geneva, Switzerland
LIRIS - Université Claude Bernard - Lyon 1
43 bd. du 11 Novembre 1918
69622 Villeurbanne, France

Abstract. Given the ongoing evolution and broadening of network environments, one should reconsider computer security from a new point of view. Indeed, the increasing transparency of network connections is a wide open door to new kinds of distributed attacks exploiting, among others, the inherent flaws of TCP/IP.
In this paper, we advocate that future Intrusion Detection and Response Systems (IDRS) should exhibit characteristics adapted to such environments. Thus, we propose an architecture of distributed IDRS inspired by natural systems. On the one side, the detection process mimics the functioning of natural immune systems: it monitors crucial computer processes and computes a deviation value allowing to discriminate between "normal" and "abnormal" behavior. A strong deviation is regarded as the sign of a possible attack. A population of mobile agents, the Intrusion Detection Agents (IDAs), which are sensitive to this deviation, are responsible for the detection of the corresponding suspicious activity. On the other side, the alert raising and response processes are based on communication mechanisms present in social insect colonies: an artificial communication medium called "artificial pheromone" is used to build alert gradients (mapped over the network and originating from threatened machines). A population of mobile agents, the Intrusion Response Agents (IRAs), sensitive to these pheromones, react to these alert gradients by implementing a distributed response process. We present the overview and principles of our architecture, as well as a detailed description of its intrinsic components. We describe some of our simulation work dealing with parameter analysis which was previously achieved. In the present paper, we discuss in greater details some new results.

1 Introduction

The continuing expansion of network connectivity forces upon us a new way of thinking the computational process. Traditional schemes are no longer valid when data housed on an Australian computer can be accessed as transparently as a local disk from a French machine. We are now witnessing a maturation process that induces the disappearance of centralized monolithic systems and the birth of a complex global distributed environment. It is now often more interesting to bring the computation to data than the opposite. Meanwhile, this evolution is not painless. As our systems are becoming more open, they are also becoming more sensitive to malicious attacks. In this paper, we are interested in exploring the benefits a mobile agent-based system can bring to the problem of intrusion detection. In a first stage, we do not approach the problem of mobile code protection. Obviously, mobile agents exhibit and withstand specific menaces based on their migratory abilities. Nevertheless we propose using this unique skill to build systems able to cope with the ever changing distributed environment of a network. Moreover, we advocate the idea that in the security domain, only a distributed framework is able to deal efficiently with contemporary computer distributed environments. The work presented in this paper is destined to exhibit several characteristics.

- The efficiency of a distributed detection system.
  Our goal is to exploit advantages of both multi-point detection probes (latency and load reduction, autonomy, dynamical adaptation to heterogeneous environments, fault tolerance) and artificial immune system characteristics (adaptive learning, self survey, standard behavior deviation, etc.) .
- The reactivity of a collective response system.
  Response behavior is implemented by exploiting dynamical information gradients mapped on the network. Each response agent only perceives these fields built during the detection process.
- The robustness and stealth of a framework based on MA technology.
  Detection and response behaviors are implemented by mobile reactive agents whose behavior include a part of randomness, reducing probabilities of interacting with potentially hostile code.

We thus propose in this paper an intrusion detection and response system (IDRS) combining on the one hand an immune system metaphor for detection, and on the other hand a social insect metaphor for response.

This paper is organized as follows. We give in the following section the requirements for an IDRS. We describe in section 3 the natural system inspiration and show its relevance to the implementation of a mobile distributed system. Section 4 describes the setting of our model. We then present in section 5 our test experiments and the results obtained, before concluding and giving some directions for future work in the last section.

2 System Requirements

The abilities that an IDRSystem should exhibit are numerous. Some of them are still out of reach of actual technologies, but they are nonetheless essential to the intended role of such a system. An exhaustive study of these capabilities was done in [1].

1. An IDRSystem must be able to run *continuously* with a minimal part of human intervention. Ideally, it should even be able to *adapt* itself without supervision.
2. It must exhibit *fault tolerance* and be able to retrieve a coherent state after an emergency stop. This stop can be accidental or malicious, the idea being to make them both impossible to happen.
3. It should be able to *monitor itself* and to detect any alteration of its components. It must then be able to purge itself to attain a sane state. Again, such an alteration should ideally be made impossible.
4. It must exhibit *some functional reactivity* to be able on the one hand to react in a timely constrained manner to any perturbation, and on the other to adapt itself to a changing security policy or a changing environment such as a changing user behavior, software or hardware evolution.
5. The resource consumption imposed by the IDRSystem should be minimal and its behavior must not interfere with legitimate user activities.
6. When the network structure changes, in size or topology, the IDRSystem must be able to continue to run efficiently and without noticeable delay. *Dynamic structural reconfiguration* of the IDRSystem must thus be possible.
7. Finally, if some of its inner components were to stop running, performance degradation should be slow and minimal.

These requirements meet the characteristics exhibited by natural systems. Indeed, these systems are able to survive and evolve in complex and potentially hostile environments by using mechanisms such as self-organization, adaptation and learning.

3 The Natural System Inspiration

Natural systems are complex systems endowed with mechanisms allowing them to react efficiently to any perturbation coming from their environment by adapting themselves to these changes. They exhibit interesting characteristics from a "computer oriented" point of view like permanent adaptation, self-organization or robustness. We propose in this paper a computing paradigm endowed with these characteristics, taking its inspiration from two kinds of natural systems: the immune system and social insects. To do so, we define the global system to be programmed (for instance a distributed system for intrusion detection and response) as an entity, evolving in a complex and potentially hostile environment. The entity has to survive by reacting and adapting to changes in the environment. Thus, the system is able to compute its state and compare it to some safe reference state. We suggest the use of: the immune system metaphor to build the Intrusion Detection System (IDSystem) and the social insect indirect communication behavior to build the Intrusion Response System (IRSystem).

3.1 The Immune System Metaphor

Overview. In the human body, the immune system can be seen as a complex network of specialized cells and organs that has evolved to defend the body against diseases and infections by foreign invaders such as bacteria, viruses, fungi, parasites, and debris. In a first step, the immune system attempts to prevent the entry into the body of external organisms. In a second step, it seeks their presence in the body to destroy them. Hence, it distinguishes between molecules and cells of the body called "self" from foreign ones called "non self". The immune defences of the body normally coexist peacefully with cells that carry distinctive "self" marker molecules. However, when immune defenders (lymphocytes) encounter "non self", they have to eliminate them quickly. Any substance which is responsible for triggering an immune response or alert is called an antigen.

Adequacy of the Immune System Metaphor to the Intrusion Detection Process. Similarly to natural systems, computer systems evolve in open complex environments such as the open internet. This makes them vulnerable because they could be subject to perturbations or attacks. Thus, computers, like natural systems, have to protect themselves. Mapping the immune system metaphor to the context of a computer network seems to provide an interesting issue. Indeed, we can identify in the IDSystem specific entities, the Intrusion Detection Agents (IDAs), which are computer defenders. Moreover, the notion of safe and non safe behavior also has its counterpart in immune systems. When IDAs detect a suspicious activity, they have to launch an alert and react to it. Thus, IDAs must examine the current state of various programs and compute a deviation with respect to a normal activity. If the deviation is higher than a certain threshold an alert is launched waiting for the response mechanism to come into play. This threshold, also called Suspicion Index (SI), depends on the tolerance level set by the security administrator for the deviation of the supervised application.

3.2 The Social Insect Metaphor

Overview: Stigmergy and Foraging Behavior. Social insects organize themselves to ensure the survival of the colony by using individual behaviors to build a cooperative collective behavior. Such behaviors can be observed from different activities: foraging, nests building or larvæsorting. Cooperation in these systems is mediated by an efficient communication mechanism relying on the inscription of task evolution in the environment. This paradigm, introduced for the first time by P. P. Grassé in [2], describes the way social insect communities (ants, termites, bees, etc.) interact through their environment. Schematically, each entity has a local view of its neighborhood but uses a chemical volatile substance (the pheromone) to mark its environment when achieving a collective task. The deposition of pheromone creates a gradient field in the environment which tends to attract other insects and to enroll them in a self-catalytic behavior. When the task is finished, no more pheromone is deposited, leading to the disappearance of this information after a period of time through an evaporation mechanism. During the foraging activity, each time an ant deposits a pheromone along its path, it reinforces the probability that other ants will choose the same path to reach the food. This paradigm has inspired many computer scientists in various research domains such as robotics, network routing and optimization algorithms [3]. In all these cases the global and complex collective behavior emerging from interactions between simple entities is dominating. Thus, building a synthetic pheromone is a good way to code control information in a distributed dynamic environment as stated by [4] and [5]. It provides different levels for tuning the control information.

Adequacy of the Social Insect Foraging Behavior Metaphor to the Distributed IDRSystem. The network is a distributed environment, subject to perturbations and dynamic evolution. The behavior of social insects seems to provide an interesting solution to deal with such an environment. Their stigmergic behavior, based on the construction of a pheromonal gradient coding the launching of an alert after an attack has occurred, provides an adaptive mechanism to support the diffusion of a control information. This requires a reactive behavior in a complex and evolving environment like a network of computers. In this context, the use of this metaphor allows MAs (artificial ants) to detect the location of an alert and go to the source of the attack in order to respond to it. In our IRSystem, the response consists in intervening where the source of the alert is located by diffusing the alert in the neighborhood of the target.

4 Mapping the Natural System Inspiration to an IDRS

This section focuses on the design goals we retained for the intrusion detection and response model. The model has been designed to allow:

– intrusion detection based on IDAs, which maps the functionalities of the natural immune system to distinguish between normal and abnormal events (respectively "self" and "non self" in the immune system) as explained in [6];
– intrusion response based on IRAs, which maps the collective behavior of an ant population by triggering throughout the network the release of a synthesized information specific to the collected events. This kind of collective paradigm is very interesting because it consists in having each ant execute a rather light task (MAs play the role of ants in the IRS) to induce collectively a more complex behavior.

One of the main points in our approach is that the IDS and the IRS are completely distributed in the network and without any centralized control. Both systems essentially consist of MAs which travel across the network, dynamically adjusting their routes according to collected events, without any simple way to trace them. Besides, our MAs are quite polyvalent because they can detect and/or respond to intrusion. This enhances the difficulty for an attacker to distinguish between IDAs and IRAs as well as it emphasizes their stealthy aspect.

4.1 The Setting of the Model

Detection Step. Our approach is targeted at corporate intranets, which correspond to logical security domains. We subdivide an intranet into several smaller local domains consisting of a set of hosts or machines. We want to avoid a monolithic IDS on every host because of its cost; instead, we propose to dispatch MAs, dynamically visiting and randomly monitoring different local domains. To detect local attacks, these IDAs, responsible for a local domain, have to be able to discriminate between normal and abnormal activity. We choose to examine the execution of different programs and compute their deviation with respect to a normal activity.

Response Step and Artificial Pheromone. The different steps of the response are the following.

1. The setting-off of an alert by an IDA. This is done by building and spreading artificial pheromone through the setting of the following fields :
   – the IDA Identifier which generated it;
   – the Suspicion Index: SI;
   – the Gradient: Gd;
   – the number of Hops: n;
   – the date of its generation;
   – the date when it was deposited at the intermediate node i.
2. The alert detection by an IRA: an IRA detects the previously diffused pheromone field and and travels to the alert source by climbing the pheromone gradient.
3. The implementation of response behavior at the alert source.

5 Simulation Description

This section exposes the simulation part of this research work. We refer the reader to [7] for more details.

The main purpose of the simulation is twofold.

– First validate the soundness of our IDRSystem, which mimics the behavior of natural systems. Indeed, issues like the behavior of a self-organized system based on single and autonomous entities to detect intrusions as well as to respond to these intrusions can be observed, thus showing that the IDAs (respectively the IRAs), completely distributed in the environment, detect on time (respectively respond on time to) a suspicious activity.
– Further assess the values of the different parameters needed for the IDRSystem implementation.[1]

A simulation tool called *Starlogo* [8] is chosen because it is an appropriate programmable modeling environment for exploring the working of decentralized systems. Thus a meshed network with 20 nodes is modeled with Starlogo.

All results are reported in the following paragraphs and are split in two parts.

– The detection part.
– The response part.

5.1 Detection Simulation

Simulation Topology. The simulation topology of the network where the detection scenario takes place, includes the following points.

– There are 20 hosts in the topology which are represented by single nodes; the nodes are numbered from 0 to 19 and are organised in a meshed network which could correspond to a local security domain.
– Each host knows only its neighbor nodes.
– Different nodes are subject to an attack.

Simulation Context. The ID scenario is briefly the following. After the network is built, IDAs[2] are dispatched randomly through the network in order to probe the different nodes. Each of them can detect a suspicious activity and locally launch an alert. As already mentioned, the IDAs[3], dispatched in the network, collect application specific system calls and compute the deviation between these system calls and other sane system calls stored in a database. The arriving system calls at a node are emulated with a sequence of bits. Based on the work done by [6] and [9], the average length of each sequence is set to 10. These system calls are also launched at irregular intervals from random locations in the simulated network. For simplicity, we decide a-priori that the sane sequence

[1] The principal parameters which were investigated, essentially relate to the pheromone evaporation and inhibition mechanism as well as the number of agents needed to make our system viable. Indeed, too many agents could also decrease the performance of the system and even be the starting-point of a DoS.

[2] In Starlgo, IRAs and IDAs are represented as single moving entities named turtles.

[3] In fact, there are different colonies of IDAs according to the type of intrusion they have to detect. This will be specified in the pheromone by the IDA identifier field that corresponds to the type of IDA which built it.

is the null sequence, i.e. each bit is set to 0. Besides, the deviation between the arriving system calls and the null sequence is computed using the Hamming distance[4].

Simulation Results. The detection simulation goal is to play with and tune the main IDSystem parameters which are:

- the number of travelling IDAs;
- the number of intrusions launched;
- the value of the Suspicion Index (SI).

A series of 15 system calls on randomly selected nodes is launched and the duration[5] needed by the dispatched IDAs to detect each suspicious sequence is computed. It is possible that more than one sequence reaches the same node.

The sequences are also randomly built. The same simulation is repeated 20 times[6] varying, for each simulation, the SI value or the number of dispatched IDAs. Thus, the system calls corresponding to a suspicious activity depend on the previously fixed SI. Indeed, the higher the SI value, the higher the probability that a system call corresponds to a suspicious activity. Figure 1 (left) shows the obtained results for SI varying from 3 to 6 and for a fixed number of IDAs set to 10. Figure 1 (right) shows the obtained results for a number of IDAs set successively to 10 and 20 and a SI value fixed to 3. In each case, the system runs until all intrusions are detected. For instance, in Figure 1 (Left) each of the 15 system calls corresponds to an intrusion for SI=6 whereas only 5 of them correspond to an intrusion for SI=3.

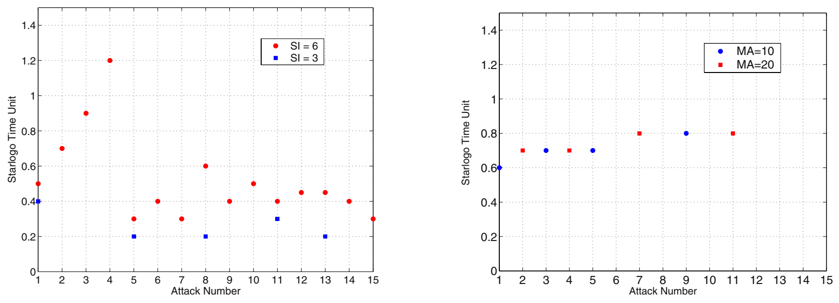


Fig. 1. Left: detection with 10 IDAs: SI=3; detection with 10 IDAs: SI=6 - Right: detection with 10 IDAs: SI=3; detection for 20 IDAs: SI=3.

[4] Let x and y be two binary sequences of the same length. The Hamming distance between x and y is the number of indexes at which x and y differ.
[5] The duration is expressed in Starlogo Time Unit (STU) for more simplicity.
[6] With the same series of system calls.

Comments and Conclusion. One can make the following observations.

- When the number of involved IDAs is fixed, actually the number of detected attacks increases when the SI value increases. Besides, in Figure 1 (left), when the SI value increases the average duration to detect an attack increases. In fact, if the number of suspicious activities is important, IDAs require more time to detect all the attacks because they have to build the electronic pheromone.
- When the SI value is fixed, the average duration to detect an attack is approximately the same with 10 IDAs or 20 IDAs. One can even observe that on average, this duration is a little higher with 20 IDAs. Indeed, it seems that in the extreme case where there are too many IDAs dispatched in the network (Figure 1 (right)), IDAs spend more time travelling from one location to another and interfering with each other, than actually performing the computation to detect the attacks. This could be due to simultaneous detection of attacks at the same location.

5.2 Response Simulation

Source Tracing. The network is a distributed environment, subject to perturbation and dynamic evolution. When an attack is detected on the network, it is not trivial to locate efficiently and rapidly its source. The behaviour of foraging ants seems to provide an interesting solution to this problem. In nature, they release in their vicinity a chemical information (the aforementioned pheromone) to trace the way from their nest to the location of some food; this metaphor is used to allow IRAs (artificial ants) to detect the location of an alert and to follow up to the source of the attack in order to answer the attack.
An alert message is initiated at a node as soon as a local IDA present at the node detects an anomaly. For this locally detected attack, the IDA creates a so-called *pheromonal message* which is randomly launched across the network and will help other IRAgents in the system to trace the way back to the alert source. The IDAgents dispatched through the network are able to launch an alert and to build and disseminate an electronic pheromonal information synthesizing the attack scenario for other IRAs. The IRAs, completely distributed in the network, can track this pheromone and travel up the pheromonal gradient back to the source. The different fields which compose the synthesized pheromone are gathered in a list as follows.

- The identifier of the IDA which detects the suspicious activity and builds the pheromone: *IdA*. In fact, there are different colonies of IDAs (and IRAs) according to the type of intrusion they have to detect.
- The suspicion index of the alert: *SI*. The proposed response scheme depends on a behavior-based IDSystem scheme depicted in [10]. In [10], IDAgents dispatched throughout the network collect application-specific system calls and compute the deviation between these system calls and other safe system calls stored in a database. In other words, *SI* is the deviation between the

"self" events stored in the database and the monitored events when the agent enters the node and detects an attack.
- The number of hops: *Hop*. It corresponds to the distance in terms of the number of links through which the pheromonal information will be propagated from the initial node.
- The pheromonal gradient: *Gd*. Like ants with respect to the source of food, IRAs have to find the better way to locate the source of an alert. To this end an electronic gradient field, *Gd*, is introduced in the pheromone. When the pheromone is diffused across the network, *Gd* diminishes hop by hop according to a strictly decreasing function. *Gd* is used in the opposite direction by other travelling IRAgents to travel up to the source of the attack.
- The date *t0*, which corresponds to the date when the attack is detected on the initial node and the pheromone is built.
- The date *ti*, which corresponds to the date when the pheromone is deposited on each intermediate node *i* during the pheromone propagation.

For the same series of 15 system calls of the section 5.1, the IDA suspicion index is set to 6 so that 5 of them are considered to be suspicious (see Figure 1). Figure 2 (left) shows the date of pheromone evaporation at each visited node for these 5 suspicious activities. Each line represents a path over which the pheromone is propagated. Table 1 corresponds essentially to the fastest pheromone evaporation among the previous simulations[7] and summarizes the IRA response frequency according to the number of dispatched IRAs[8]. The simulation is repeated 10 times[9].

For the fastest pheromone evaporation (path number 3), the date of arrival of IRAs[10] on the different nodes of the network are collected in Figure 2 (right). First time, second time and third time correspond to the number of times an IRA reaches a node at the corresponding date. The curve represents the date of the pheromone evaporation. From this Figure, we see that there are 3 responses on time at the node 5, 1 at the node 6 and 2 at the node 11. One can notice that there are more IRA arrivals than the maximal number of dispached IRAs. This is due to the fact that some IRAs have time to move from one location to another during the fastest evaporation.

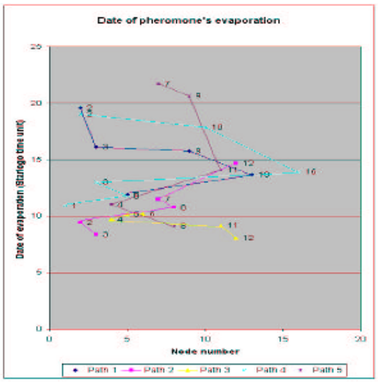
In the last series of simulations, each time a new IRA reaches a node with a pheromone, the pheromone evaporation is speeded up thanks to the inhibition index, as it is explained in a previous work published in [11]. Figure 3 (left) shows the date of the IRA arrivals during the fastest evaporation duration with

[7] This is the yellow path number 3 in Figure 2 (left) where the suspicious activity happened at the node 6 since it is the last node where the pheromone evaporated.

[8] In this experiment, there is just one type of IRAs corresponding to one type of pheromone.

[9] Obviously, for each of these 10 simulations, the pheromone was artificially deposited along the path number 3 following the same series of nodes and evaporating at the same dates.

[10] The number of IRAs dispatched in the network is random and varies from 5 to 10 in Table 1.



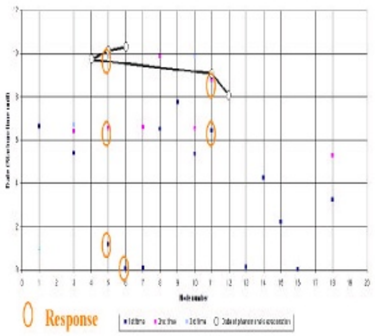


Fig. 2. Left: pheromone evaporation dates for a 5 hop simulation - Right: IRAgents responses during the fastest evaporation.

Table 1. Response frequency according to the number of IRAs.

Simulation	Answers	Agents	Frequency
1	6	10	0.6
2	6	10	0.6
3	3	5	0.6
4	4	8	0.5
5	7	10	0.7
6	8	9	0.889
7	4	7	0.571
8	4	10	0.4
9	5	7	0.714
10	2	9	0.222
Average frequency	4.9	8.5	0.580

the inhibition index set to 80%. Table 2 (left) summarizes the corresponding response frequency with a number of IRAs varying from 7 to 12.

Figure 3 (right) shows the date of the IRA arrivals during the fastest evaporation duration with the inhibition index set to 90%. There is no answer on time in this case. Table 2 (right) summarizes the corresponding response frequency with a number of IRAs varying from 3 to 6.

Comments and Conclusion. In almost all simulations there are IRAs which answered before complete evaporation of the pheromone. Without inhibition, considering that the number of acting IRAs during the evaporation duration is limited to 10, there is on average 58% of the dispatched IRAs which respond on time. Besides, there are on average 4.9 effective responses. These first results show the efficiency in using the concept of electronic pheromone to trace the

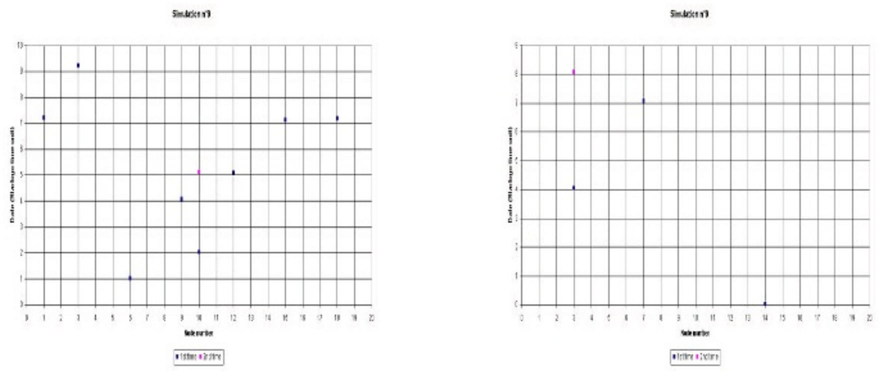


Fig. 3. Left: agent response dates with 80% of inhibition - Right: agent response dates with 90% of inhibition.

alert source and to obtain a significant number of responses. This also proves that the theorical values computed in [11] are viable.

With inhibition, the influence of the inhibition index on the number of responses is notable. On average there are 16,7% of the launched IRAs that respond on time to an intrusion with the inhibition index set to 80% and 14,3% of the launched IRAs that respond on time to an intrusion with the inhibition index set to 90%. Compared to the previous tests without any inhibition index, this second set of tests validates the inhibition index effect in the sense that it considerably decreases the number of responses. Hence, it could be used to control the number of responding IRAs. This is quite powerful for the design and the tuning of the IRSystem because it avoids having too many IRAs converging to the same source of alert. This control mechanism is also needed to allow the IRAs to be permanently dispatched throughout the whole the network.

The simulation is convincing in the sense that it legitimizes the idea of embodying the IDRSystem in a population of single entities, namely IDAs and IRAs. Both agent populations react suitably exhibiting a global detection and response behavior emerging from a local view of their neighborhood. Finally, the network (nodes as well as links) is an enormous potential to vehicle the inter-agent information necessary to build a dynamic and furtive IDRSystem completely distributed. However, the simulation alone does not suffice because MAs (and of course the IDRSystem itself) are not placed in a real environment and a simulation tool such as Starlogo does not take into account all the real-world constraints inherent to an authentic computer network. Thus, the next obvious step after the simulation is the IDRSystem implementation.

6 Conclusion and Future Work

In this contribution, we addressed the question of security in computer networks by proposing a distributed Intrusion Detection and Response System (IDRS)

Table 2. Left: response frequency according to the number of IRAs: inhibition index set to 80% - Right: response frequency according to the number of IRAs: inhibition index set to 90%.

Simulation	Answers	Agents	Frequency
1	0	7	0.000
2	2	8	0.250
3	2	7	0.286
4	2	7	0.286
5	2	9	0.222
6	1	8	0.125
7	0	8	0.000
8	1	8	0.125
9	2	9	0.222
10	2	12	0.167
Average frequency	1.5	8.3	0.168

Simulation	Answers	Agents	Frequency
1	2	3	0.667
2	0	5	0.000
3	2	5	0.400
4	0	3	0.000
5	1	6	0.167
6	0	4	0.000
7	0	6	0.000
8	0	5	0.000
9	0	4	0.000
10	1	5	0.200
Average frequency	0.6	4.6	0.143

based on two natural systems : immune systems and social insects colonies. We presented an overview and principles of its architecture, as well as a description of its intrinsic components. The important features of our IDRS concern its distributed nature, since there is no centralized control. Mobile agents travel the network having only a local perception of their environment. Their behavior is based on the local information they collect. Being highly mobile and exhibiting random-based behavior, they are difficult to interact with and thus to trace. This also contributes to the robustness of our IDRS. We validated the model of our IDRS by running a simulation. The agents displayed a correct behavior in the sense that the intrusion detection agents (IRA) detected the launched attacks and the response agents (IRA) went up the pheromone trail to the source of the attack before the pheromone actually evaporated. We also empirically determined, for a fixed network topology and a given suspicion index, the optimal number of agents. We also presented the detailed architecture of our IDRS. The IDAs and IRAs have been implemented as well as the services involved. This was done using Sun Java technology and the JSEAL-2 framework. The tested intrusion scenarii ascertained the soundness of the design.

Our present work concentrates on studying control mechanisms to manage agent populations. Our future work will investigate several aspects. First, we will address the problem of mobile code security and related tools like authentification, cryptography and right management. Then, we will focus on the use of data mining techniques to analyze log files in order to generate profiling data. Our ambition remains to apply our IDRS in a real environment such as the intranet of the Centre Universitaire Informatique of the University of Geneva. In the long term, we plan to explore adaptation mechanisms from artificial life fields like genetic programming and complex systems, as well as extensions like the influence of wireless networking environments on the security problem we are addressing here.

7 Acknowledgments

Thanks to Edwin Cox for his contribution to the implementation.

References

1. M. Crosbie and E. H. Spafford. Defending a computer system using autonomous agents. pages 549–558, 1995.
2. P.P Grassé. La reconstruction du nid et les interactions inter-individuelles chez les bellicoitermes natalenis et cubitermes, la thorie de la stigmergie - essai d'interprtation des termites constructeurs. *Insectes Sociaux, no. 6*, pages 41–81, 1959.
3. E. Bonabeau, M. Dorigo, and G. Theraulaz. Swarm intelligence: From natural to artificial systems. *Oxford University Press, New York*, 1999.
4. H.V.D. Parunak and S. Brueckner. Entropy and self-organization in multi-agent systems. *Proceedings of 5th International Conference on Autonoumous Agents,Montreal, Canada*, May 2001.
5. P. Valckenaers, M. Kollingbaum, and C. Zamfirescu H. Van Brussel, O. Bochmann. The design of multi-agent coordinationand control systems using stigmergy. *Proceedings of 3rd International Workshop on Emergent Synthesis (IWES'01), Bled, Slovenia*, March 2001.
6. S. Hofmeyr and S. Forrest. Architecture for an artificial immune system. *Evolutionary Computation 7(1), Morgan-Kaufmann, San Francisco, CA, pp. 1289-1296*, 2000.
7. David Landecy. Simulation d'un modle de dtection et de rponse une intrusion inspir des systmes naturels. Master's thesis, Centre Universitaire d'Informatique, 2002.
8. Starlogo. http://el.www.media.mit.edu/groups/el/projects/starlogo/.
9. S. Forrest, S. Hofmeyr, A. Somayaji, and T. Longstaff. A sense of self for unix processes. 1996.
10. N. Foukia, D. Billard, and Pr. Juergen Harms. Computer system immunity using mobile agents. HPOVUA Workshop, Berlin, June 2001.
11. N. Foukia, S. Hassas, S. Fenet, and J. Hulaas. An intrusion response scheme: Tracking the alert source using stigmergy paradigm. July 2002.

Secure Mobile Agents Applications: Implementation through Platform Services

Joan Ametller, Sergi Robles, and Joan Borrell

Computer Science Dept. Universitat Autònoma de Barcelona
08193 Bellaterra, Spain
Joan.Ametller@uab.es

Abstract. In this paper, we provide a solution for the implementation of protection mechanisms using platform services. Implementation of protection mechanisms is the main obstacle of secure mobile agent applications. A cryptographic service can allow agents to self protect their code and data. The implementation of such a service is far from trivial. We provide some schemes for programming this implementation. We have implemented this cryptographic service in the MARISM-A/JADE platform, using Java. Any application using mobile agents can use our mechanisms to implement agent protection.

1 Introduction

Advantages provided by mobile agent technology make it indispensable for some applications, such as Sea-of-Data applications [1]. Mobile agent applications have some requirements hard to satisfy, usually regarding security.

Agents will be executed on platforms, following an itinerary. Platforms control mobility critic processes, such as migration, dynamic code loading, data protection, or agent integrity validation. Most of the security of an application relays on these platform processes. On many secure applications, code and data must be kept secret for all platforms except for the intended one. There exists cryptographic mechanisms to perform this, but implementing them in mobile agent environments is complex. Implementing these mechanisms usually implies to set a very rigid agent taxonomy, adding complexity to the process of restarting agents after the migration transit.

In this paper, we present a guideline for the implementation of flexible protection mechanisms based on cryptography. Firstly, we solve the problem of accessing security related mechanisms by means of a service architecture. This allow both, cryptographic functionality for agents (encrypting, signing, integrity verification, etc.) and security information accessing (directory with identities, certificates, etc.). Secondly, we provide some techniques to avoid common attacks to these structures.

The rest of the paper is structured as follows. Second section describes in detail the common scenario of secure mobile agents applications. Third section analyzes some protection mechanisms, whilst next section is devoted to the specific issues regarding the implementation of this mechanisms on a concrete platform. Finally, conclusions are exposed.

2 Scenario

Techniques for agent integrity, privacy and authenticity are usually based on cryptographic mechanisms [1], [2] [3], [5]. These methods usually wrap code in such a way that it is handled as data. This allows to share same mechanisms for both code and data. Main differences between different methods performing this are due to the structures used to store the contents to be protected and where and how cryptography is exactly used. Furthermore, digital signature and hash functions based mechanisms are also used, among others. Most of the applications requiring this type of schemes share a common scenario. Analyzing security and providing solutions for the specific problems in this scenario is therefore essential to obtain a general approach.

Main requirements of this kind of scenarios assume the existence of a public key infrastructure (PKI), and a pair of asymmetric cryptographic keys in every platform. Keys are mainly used to protect data and code, hiding them in such a way that only intended platforms are able to access them.

When protecting mobile agents with mechanisms such as those described above, the platform usually drives the process of extracting protected data from the agent. This is not the only way of performing this. Agents can also carry out this task. Thus, there are two approaches: platform driven and agent driven.

In the first case, the platform must know what is the structure of the agent and how it is encrypted. Therefore, it can decrypt the agent and restore its execution. Moreover, there can be other processes in the platform like signature verification or alternative protections.

The problem of this approach is that agents are very rigid if all cryptographic mechanisms for data protection are controlled by the platform. If the implementation of the scheme becomes inefficient, or an update is required, all platforms must be modified and old agent code must be rewritten in order to support the new implementation.

The second case, agent driven protection, is an opposite methodology that allows to create protection mechanisms that are controlled by the agent. A way of implementing this new approach is to introduce security mechanisms within agent architectures ([2]).

2.1 Agents Based on Architectures

An agent architecture defines a concrete agent taxonomy and the code necessary to interact with internal agent structures. The goal of this approach is two folded. Firstly, to move the handling of agent's internal structures, normally performed by the platform, into agent's code. The main advantage of moving this process provides the possibility of supporting multiple agent taxonomies in the same platform. Secondly, to develop the architectures framework in such a way that it allows a high reusability of the architectural code. Therefore, developing an agent from a certain architecture does not become a tedious task.

Using architectures for implementing agent protection systems is somewhat difficult considering an scenario such as the one described above. The main advantage of using agent architectures to perform this task is that agents with totally different internal structures can run simultaneously in the same platform. The main drawback is that the platform doesn't know anything about the internal structure implemented on every agent. Therefore, encryption mechanisms can only be used from the architecture they are defined in. The platform cannot perform this decryption because it cannot locate the structures, neither know their nature.

In order to implement security systems based on cryptography, architectures must have access to platform's private key or, which is the same, to a decrypting function using that key. This is obviously a serious problem. If an agent is able to access to this decrypting function at any platform, then it could decrypt data which is intended for any platform.

We have chosen this last agent driven approach of implementing protection mechanisms as a concrete scenario. Particularly, the rest of the paper will use the MARISM-A platform as an example.

2.2 MARISM-A

MARISM-A (an Architecture for Mobile Agents with Recursive Itineraries and Secure Migration) is based on JADE [7]. JADE is a well known software framework programmed in Java to simplify the implementation of multi-agent systems. It observes most of the FIPA specifications [4] for agent communication and platform elements, and so does MARISM-A. Furthermore, JADE is licensed under the LGPL (Lesser General Public License) that allows its modification.

We consider agents are pairs (C, D) where C is the code of the agent and D its data. C basically contains the architectural code. All architectures share the same method for starting agent's execution. D are data being born by the agent. They can be split as (D_1, D_2) where D_1 are agent data and D_2 is agent code wrapped as data. An agent architecture uses cryptographic methods for extracting the encrypted code from D_2 and afterward it executes this code.

In the next sections we will expose how the security mechanisms implemented in MARISM-A are used to protect agents based on this architectures. A security analysis of this methods will be also done, explaining what kind of attacks are possible and which mechanisms are used to prevent them.

3 Protection Mechanisms through Services

The basic problem of agent driven protection mechanisms was that agents needed to access platform's private key. The proposed scheme uses a public decrypt function that indirectly uses this key. This function has some verification mechanisms to ensure that it is not used by malicious agents trying to obtain data from other agents.

FIPA specification regarding the abstract architecture of platforms [9] proposes a model for services. We use this concept of services for the implementation of protection mechanisms. Specifically, the decrypt function exposed above is used by the agents through a service.

The basic requirement at platform level is that the platform provides a "cryptographic service". This entity can be modeled from many different ways. Moreover, a "directory service" is used to store some needed information.

3.1 Directory Service

The basic necessity for a directory service is two folded. Firstly, in every mobile agent platform which is FIPA compliant it is needed a name resolution system. In FIPA specifications [4] agents are referred through an agent identification string called AID (Agent IDentifier). This identifier is composed by some fields, such as the name of agent GUID (Global Unique IDentifier) and the transport address associated with the MTP (Message Transport Protocol) of the platform where the agent is running. Interaction between agents from different platforms it is not possible unless one of them known the transport address of the other. A directory service allows to register all the transport addresses associated to agents in the system. Agents and platforms can use this service to know in run time the address of other parties. Secondly, when using a PKI like in the described scenario, it is almost unavoidable to require a directory facility to store public keys.

3.2 Cryptographic Service

The cryptographic service is intended to be the tool that agents must use in order to legally decrypt data. The main goal we want to achieve is this: Let $E_{i,k_j}(m)$ the result of encrypting the message *m* by entity i, using the public key of platform j, k_j. What we want is that only agents created by entity i has access to contents m within platform j. As we have introduced above, we want to avoid data to be decrypted by other agents but for its owner.

As we can see, neither platform public key k_j, nor the corresponding private key, are used by the platform for itself. They are simply used by agents for protecting data, which can only be accessed within a concrete platform.

The cryptographic service is an entity that belongs to a platform that can be invoked by the agent. This entity is the only one that will have access to the private key of the platform, keeping it hidden. On the other hand, it provides a public decryption function that is a bit special. Only a specific encrypted data format is accepted: *(m,x)* pairs where *m* are data to be encrypted and *x* is an identifier that authenticates the code using them. The next pseudo-code shows how this function works:

```
data decrypt( data ) {

    data_contents = decryption( data, private_key );

    if( data_contents is not a pair (m,x)) {
        throw an error, destroy data; }

    else  if (x authenticates the querying agent) {
             return m; }

          else { throw an error, destroy data; }
}
```

As it can be seen in the algorithm, when the cryptographic service is used to decrypt some data these data are decrypted but not returned unless *m* authenticates somehow the caller. Doing this we ensure that decrypted data are delivered only to the code for which they are intended to.

3.3 Design of the Protection Mechanisms

The proposed protection scheme not only depends on the cryptographic service. In this section we describe the complete structure used in the protection.

When agent execution is resumed after a migration, the architecture code (in C) is used by the agent to extract its own code and data (D) to start the agent's task in the current platform.

The first restriction of our scheme is precisely the encryption of agent's data (D), as it has been introduced in previous section. Assuming that data m are going to be encrypted for platform *i*, the usual process is to create $c = E_{k_i}(m)$. However, in our approach we use a pair $m' = (m, x)$ for the data to be encrypted, resulting in $E_{k_i}((m, x))$ when they are encrypted. *x* is the result of applying a hash function on *C*, the code of the agent.

If we take a look to the algorithm introduced in the description of the cryptographic service, we see that a mandatory argument of the decryption function is in the form $E_{k_i}((m, x))$. The cryptographic service decrypts the data, extract hash *x* and verifies it using the hash of the caller agent. If the hash is the same, decrypted data *m* are delivered. If we analyze this mechanism, we can see that being able of replacing *x* implies to have broken cyphering. In the other hand, when the cryptographic service has rightly verified *x*, it is sure that data are delivered to the code (C) for which they have been encrypted: only owner of *m* is able to create $E_{k_i}((m, x))$.

Analyzing this mechanism, we can see a first type of attack which it prevents. We could think in attacks that attempt to modify the architecture code (in C) that is responsible to extract the ciphered content of the agent. As we can see in Figure 1, the aim of this kind of attacks could be to decrypt the data and code inside the agent and send it to some place over the network. If the ciphered data is constructed using the mechanism described above, these attacks does not succeed because the cryptographic service will check if *x* inside the ciphered content equals the hash applied to the agent who has made the call. If the code has been modified, the two values will not be equal.

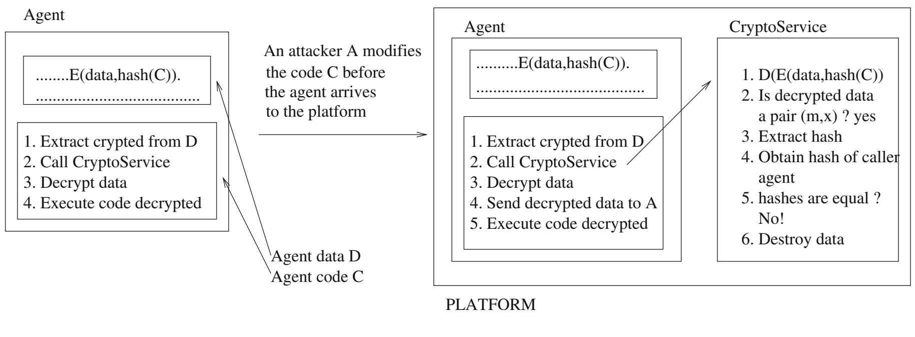

Agent
........E(data,hash(C)).
....
1. Extract crypted from D
2. Call CryptoService
3. Decrypt data
4. Execute code decrypted
An attacker A modifies the code C before the agent arrives to the platform
Agent data D
Agent code C
Agent
..........E(data,hash(C)).
....
1. Extract crypted from D
2. Call CryptoService
3. Decrypt data
4. Send decrypted data to A
5. Execute code decrypted
CryptoService
1. D(E(data,hash(C))
2. Is decrypted data a pair (m,x) ? yes
3. Extract hash
4. Obtain hash of caller agent
5. hashes are equal ? No!
6. Destroy data
PLATFORM

Fig. 1. The encrypted hash of the code prevents code modification attacks

The provided mechanism protects data from agent's code, especially from being accessed from a malicious code. This is because a hash is used to relate data with the code allowed to deal with them. Moreover, it is needed to think about another protection which is the opposite case: protecting code from malicious data.

Our architecture nature allow certain type of data (D_2) to be code that will be executed at some time. It is important to notice the case where some data injected to the agent with the right format $E(m, x)$, with a code coherent hash *x*, is really some malicious code.

As it can be inferred from this last consideration, this case is totally dependent on the architecture. It is possible to design an architecture where malicious data injection is not possible. We will discuss it and we will provide a solution for attacks based on malicious code wrapped inside the agent data.

The method below is a concrete example of how an agent can protect itself against data that could have been included inside it. We describe the mechanism applied to an agent architecture of the scenario described in section 2.

In our example, the agent uses a simple architecture using cryptography to protect data. The agent has a vector containing all the code that will be executed on every platform:

$$V = [E_{k_1}(C_1, hash(C)), E_{k_2}(C_2, hash(C)), \ldots, E_{k_n}(C_n, hash(C))]$$

C_i is the code to be executed on platform i. E_{k_i} is a function that uses public key of platform i, k_i, to encrypt data in such a way that only platform i will have access to it. C corresponds to the agent architectural code as we have explained above. When the agent migrates to a platform, the code corresponding to this platform is extracted, decrypted using the Cryptographic Service and executed by the agent.

It is obvious that security is compromised here: someone could replace any of the positions of the code vector V with some malicious code wrapped as data. The agent would execute that code without noticing the deception. This happens because the structure $E_{k_i}(C_i, hash(C))$ can be created by everybody from a code and the public key of a platform k_i. This is a serious problem: the malicious code could be used to decrypt data intended for any platform. This problem occurs because our Cryptographic Service compares the hash inside the encrypted content, with the hash of the architectural code C, which in this case has not been modified.

Because of these reasons, protecting data from malicious code is as important as protecting code from malicious data. In the example above we add a special code in the architecture part of the agent that will safely extract code and data for the intended platform.

When the agent is created, the programmer must generate a pair of asymmetric keys (pk, sk) that can be used to sign and verify data. Once these keys are generated, the public key is placed in the code of the agent (as a constant).

When the architectural code has been generated, the rest of the agent is encrypted. To carry out this operation the public key of each platform is used to encrypt the specific code and data that will be used there. This process is done as follows.

A pair $(s(C_i), C_i)$ will be created for data intended to platform i, where $s(C_i)$ is a hash of the data signed with the private key sk. Once this pair is created it will be encrypted using the method described above, obtaining $E_{k_i}((s(C_i), C_i), hash(C))$.

The code of the architecture that carries out data and code extraction follows these three steps:

1. Call the platform Cryptographic Service querying the decryption of the structure $E_{k_i}((s(C_i), C_i), hash(C))$. If the process succeeds it will return $(s(C_i), C_i)$.
2. Before extracted data is used, $s(C_i)$ must be verified against C_i using public key pk which is stored within the architectural code.
3. If this verification succeeds too, extracted data can be used. If this data is agent code, it will be executed. Otherwise, if the verification fails agent execution will be aborted and some alternative code will be performed.

Described this mechanism, let us see what happens if some malicious data was injected inside the agent. Suppose that we have an agent composed of code (C) and data (D). Some malicious entity could capture the agent while it is migrating between platforms and inject some data inside D as we have described above. When the agent arrives to the platform, the architectural code, will extract the injected data, that could be wrapped malicious code. In this case, when the architectural code attempts to extract the data it detects that it has not been signed by owner of the agent and aborts execution.

A combined attack could be to inject data inside the agent and to modify the agent code. The modification of the code could be made to force the architecture to skip the verification of the data signature extracted after the decryption process, or to modify the public key. This kind of attack would not succeed because if the code is modified, the agent will not be able to decrypt the original data using the Cryptographic Service.

Note that this protection mechanism depends totally on the architecture, and it does not affect platform code. Therefore, an architecture protected against malicious data injection does not require to implement this protection mechanism.

4 Implementation Issues

Some mechanisms have been introduced in previous section. These mechanisms provide a generic protection scheme for mobile agents based on cryptographic tools. The description was not focused on a certain implementation in order to avoid restrictions. However, new problems could arise when considering specific implementations of these mechanisms.

MARISM-A [6] (introduced in section 2) is a secure mobile agent platform based on JADE. MARISM-A includes cryptographic and directory services that are used to provide the protection mechanisms and functionality described previously.

In this section we will describe how a service infrastructure has been added to this platform and how it can be used by agents. Some implementation aspects regarding the cryptographic and directory services are addressed in minute detail due their relevance to the resulting protection mechanisms.

4.1 Service Infrastructure

The basic concept of providing a set of software services to agents that can be accessed through an infrastructure is already proposed in the FIPA abstract architecture [9]. In this specification there is a root service called *Service Directory Service*. This service is used for discovering other services within the platform.

Providing software services to agents is no a usual task of platforms. Platforms mainly provides an execution environment where agents can interact. However, it can be desirable that sometimes a platform could offer certain services in order to access exclusive resources. This can be also useful when a proxy agent acting as an interface is not feasible.

In order to have general purpose software services for agents we firstly need a service infrastructure. This infrastructure allows service extendability and provides implementation independence. Agents access to this infrastructure through a static class called *ServiceProvider*. This class is initialized when the platform is started.

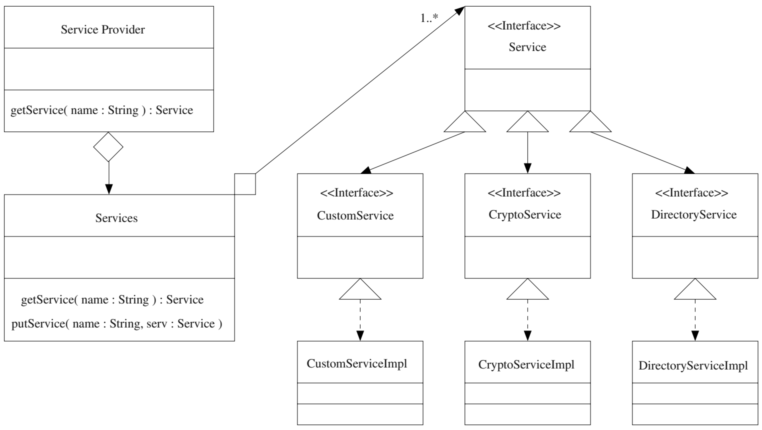


Fig. 2. Service Infrastructure

Services offered by a platform have to be programmed as classes implementing interface *Service*, which is found in the same platform. These classes implements the service by defining methods with the desired functionality (figure 2). Some basic service interfaces, such as the cryptographic service one, are offered by the platform itself. Agents only use interfaces to get the required services. This abstraction allow platforms to offer different implementations of the same service, resulting this unnoticeable for agents.

As shown in figure 2, all references to initialized services are stored in the class *Services*, which is itself contained in the class *Service Provider*. The static method *getService(Service name)* of class *Service Provider* is invoked by agents to get a reference to a concrete service. This reference will be used afterward to use the service.

Below, we describe the two main services regarding protection mechanisms that are included in platform MARISM-A, the directory service and the cryptographic service.

4.2 Directory Service

Directory service interface is used to transparently access this service. The directory service used in MARISM-A us based on LDAP [8]. In fact, the service is a wrapper to access an external LDAP server. This server is highly scalable, which is an important feature desirable in a multi agent system. Replication on several hosts makes it difficult to get this server collapsed with queries.

Every platform in a Multi Agent System (MAS) have some entries stored in the directory. Each platform entry has information regarding transport addresses and certificates associated with their public keys. One of the main advantages of this implementation is that the hierarchical structure of LDAP servers allow an easy representation of a Public Key Infrastructure (PKI). It is also possible to include the certificates of the certification authorities issuing certificates of platforms.

Storing transport addresses of platforms in the directory is also very important. This is because FIPA does not provide a direct mechanism for translation of names of platforms and transport address. This could be achieved establishing a Directory Facilitator (DF) federation including all platforms in the MAS. However, the functionality provided by this federation should not be the concern of the DF, forcing a not envisaged artificial utilization. Translating names and addresses through the directory service is a better solution.

The directory service class implements the directory service interface. This class wraps all queries it receives from agents to the LDAP directory server.

4.3 Cryptographic Service

The access to this service is also designed as an interface hiding the actual implementation. The main method of this service decrypts data for agents using the private key of the platform. In section 3 this method was deeply analyzed.

The first specific issue in the implementation consists of establishing the identity of the agent that has invoked the decrypt method of the cryptographic service. Using the proposed service architecture, agents can obtain a reference of the service and invoke any method. It is far from trivial to get a reference to the code of a calling agent from inside the decrypting method.

Our approach to solve this implementation problem consists of several parts. The main component is a table that is created when the platform starts. This table contains agent identifiers / agent code hashes pairs. When an agent is created, or when it arrives from another platform due migration, a random number is generated from a wide rang. This number, that will be the agent identifier, is given to the agent and stored into the table. The result of a hash function applied to the agent is also stored in the table. Agents have no access to the contents of this table. When an agent wants to decrypt some data, it invokes the decrypting method. As arguments to this method, it will use the encrypted data and the identifier that was assigned to it by the platform. Thus, the Cryptographic Service has a direct correspondence between the agent performing the invocation and its hash, whereby verification will be possible.

Security of this mechanism depends somehow on the random number generation and length. Also, it depends on the restrictions of the Java language. Even though, this is a solution that can be applied to many implementations. It solves the problem of securely determining who is invoking a method, and accessing to its code, from the method itself.

5 Conclusions

In this paper, we have proposed a new solution for the protection of mobile agents on a specific scenario common to most secure MA applications. In this scenario agents control their own code and internal structures by using architectures. Our solution combines architectures advantages with services provided by the platform, offering a flexible protection scheme. All the implementation are being tested on the MARISM-A/JADE platform. Using our scheme, applications using secure mobile agents are much easier to develop. Future research on this field is envisaged though the definition of new architectures using all this mechanisms, and the deployment of new applications.

Acknowledgments

This work has been partially funded by the Spanish Government Commission CICYT (TIC2000-0232-P4), and Catalan Government Department DURSI (200-SGR 00219).

References

1. S. Robles. Mobile Agent Systems and Trust, a Combined View Toward Secure Sea-Of-Data Applications. Phd Thesis, Universitat Autònoma de Barcelona, 2002.
2. S. Robles, J. Mir, J. Ametller and J. Borrell. Implementation of Secure Architectures for Mobile Agents in MARISM-A. In Mobile Agents for Telecommunication Applications (MATA). Vol. 2521 of Lecture Notes in Computer Science, pp. 182–191, Springer Verlag, October 2002.
3. Roth, V. Empowering mobile software agents In Proc. 6th IEEE Mobile Agents Conference 2002. Vol. 2535 of Lecture Notes in Computer Science, pp. 47–63, Springer Verlag, October 2002.
4. FIPA. Foundation for Intelligent Physical Agents. `http://www.fipa.org`
5. SEMOA. The SEcure MObile Agents Project. `http://www.semoa.org`.
6. MARISM-A: An Architecture for Mobile Agents with Recursive Itinerary and Secure Migration. `http://www.marism-a.org`.
7. JADE, Java Agent DEvelopment Framework. `http://jade.cselt.it`.
8. RFC 1777. Lightweight Directory Access Protocol. The Internet Engineering Task Force. `http://www.ietf.org`. March 1995.
9. FIPA Abstract Architecture Specification. Foundation for Intelligent and Physical Agents, 2002. `http://www.fipa.org/specs/fipa00001/`
10. FIPA Agent Management Specification. Foundation for Intelligent and Physical Agents, 2002. `http://www.fipa.org/specs/fipa00023/`

Protecting Mobile Agent Itineraries

Joan Mir and Joan Borrell

Computer Science Dept., Universitat Autònoma de Barcelona
Edifici Q - 08193 Bellaterra (Spain)
Tel: +34 93 5811470, Fax: +34 93 5813033
e-mail: {jmir,jborrell}@ccd.uab.es

Abstract Mobile agents are believed to be playing an important role in future e-commerce systems, offering great flexibility and improved performance. Mobile agents are processes which can autonomously migrate from host to host. The migration path followed by an agent can be abstracted for programming convenience into an itinerary. A flexible structure of itinerary is used in *Concordia* or *Ajanta* Agent Systems, using sequence, alternative, and set entries.
On the other hand, security is a fundamental precondition for the acceptance of mobile agent systems. Several itinerary protection protocols have been presented for each kind of entry. This paper improves previous solutions proposing a unique general protocol using petri net modeling, providing minimum route information to visited hosts, and strong protection against tampering and itinerary analysis attacks.

1 Introduction

In many scenarios, mobile agents (m.a.) represent customers, salesmen or mediators for information, goods and services. So, m.a. are becoming more important for Internet based electronic markets. A m.a. or mobile code consists of code and its current configuration, including global data structures, stack, heap and control information like the program counter, and it is expected to migrate between hosts that offer environments in which the code can be executed [1].

A m.a. architecture consists of agents and places. A place receives agents and is an environment where an agent executes an action. So far, different architectures have been proposed for implementations. Research labs as well as companies develop java-based systems for m.a., such as Odyssey [2], Voyager [3], Concordia [4], Ajanta [5] and IBM aglets [6].

Itineraries may be viewed as a specification or plan of the agent movements. A community of agents is considered to implement the movement plan correctly, if their observed behaviour corresponds to the plan. Following an itinerary, fixed at the beginning, the m.a. goes through a set of hosts to achieve different tasks.

An itinerary is composed of different kinds of entries. The basic form of an entry e_i is formed by a tuple $e_i = (host_i, data_i\ +\ methods_i)$, where $methods_i$ are the name of the methods of the agent's code that are invoked at $host_i$.

In order to create a flexible framework, entries can recursively contain further entries, defining a behaviour. When a set of entries has to be visited following a specified order, a sequence itinerary is created. If the order is not important, a set itinerary is defined. And finally, if only one entry has to be visited from a list of entries, an alternative itinerary is created.

We assume that hosts untrust each other. Therefore, they might try to modify the agent or to gain knowledge about its itinerary to favor themselves to the detriment of the rest. A number of protection schemes have been devised to protect certain aspects of m.a. against malicious host, yet most of them are very restricted or fail to be applicable in a general setting. We describe a protocol that falls into the category "restricted but applicable" [7].

In previous itinerary protection solutions [8,9,10] only sequence itineraries are protected, and later in [11] a proposal is presented using three different protocols (one for sequence, other for alternative and last for set itineraries) which cannot protect any construction. In this paper, we improve previous solution making use of petri net modelling and cryptographic tools in order to protect any flexible itinerary from modifications in a unique general protocol.

The protocol satisfies established security requirements in previous solutions [12,9,10,11,13], resists against "cut & paste" attacks [14], and reduces route information provided to hosts. Moreover, our protocol allows hosts to detect modifications of the original itinerary. This protocol is used in *MARISM-A* agent platform [15] to protect their agent itineraries.

The rest of the paper will proceed as follows: In Section 2, we define the flexible itinerary framework. We discuss previous works to protect itineraries against malicious attacks in Section 3 and petri nets are illustrated in Section 4. Later on, in Section 5, we present our proposal where security properties are analyzed. Finally, in Section 6, we sum up the results of this paper.

2 Flexible Itinerary Framework

The concept of itinerary was introduced in the Concordia system [4], and improved in [16]. It describes a set of locations to which a mobile agent is going to travel and the work to be accomplished at each location. A location is a host with agent system support. An itinerary is composed of different kinds of entries. The basic form of an entry e_i is formed by a tuple $e_i = (host_i, data_i + methods_i)$, where $methods_i$ are the name of the methods of the agent's code that are invoked at $host_i$. Some entries recursively contain further entries:

- A **Sequence** is a list $[e_1, \cdots, e_n]$ of entries with $n \geq 1$ defining that the nodes specified by entry e_i $(1 \leq i < n)$ must have been visited before the nodes of entry e_{i+1} are visited.
- An **Alternative** is a list $< e_1, \cdots, e_n >$ of entries with $n > 1$. Only one entry is visited from the whole set of entries. The entry is chosen by the m.a.
- A **Set** is a list $\{e_1, \cdots, e_n\}$ with $n > 1$ specifying that the elements $e_1, \cdots, e_n$ are handled in any order as long as each element is visited exactly once.

If the complete migration path of the m.a. is known in advance, the owner can insert appropriate entries into the agent's itinerary before dispatching it [5].

On the other hand, other solutions show that the agent is able to migrate to a yellow pages server, where it will search the next destination.

All the locations are not necessarily trustworthy. The hosts might try to extract sensitive information from the agent or change the behavior of the agent by removing, modifying or adding to its data and code. So, flexible itinerary frameworks are vulnerable to attacks. We define a set of general security properties that any protocol has to satisfy in order to protect the itinerary [12,9,10,11,13]. The properties are the following:

P1 Integrity: It should be infeasible for a host to modify the m.a. itinerary (including recursive entries). The agents shall be protected such that they can acquire new data on each host they visit, but any tampering must be detected by other hosts on the agent's itinerary.

P2 Confidentiality: No host should be able to obtain/capture the particular data of other hosts.

P3 Forward Privacy: In a sequence itinerary, we should reduce the information exposed. In such a way, each host is only informed about its predecessor and successor. Remaining route information is hidden in order to prevent itinerary analysis and reveal communication relationships. This property is infeasible with alternative and set entries.

P4 Verifiability: Every host should be able to verify that it forms part of the initial route generated by the agent's owner.

P5 Source verification: Every host should be able to verify that the mobile agent actually comes from previous coded host in the agent route.

P6 Exactly Once: Every host should be able to detect replay attacks. The host has to detect if the present m.a. is a new m.a. or a copied m.a.. It also detects "cut & paste" attacks with a portion of one m.a.

P7 Single Entry : Only one host of the list of all specified entries in an alternative itinerary has to be visited.

3 Related Work

A number of protection schemes have been devised to protect certain aspects of m.a. against malicious host, yet most of them are very restricted or fail to be applicable in a general setting. Different mechanisms allow the agent's owner to check the integrity of the mobile agent, after the agent migration. Others are focused on maintaining the secrecy of the agent's computations and data.

[17] presents a mechanism based on execution tracing and cryptography that allows to detect attacks against code, state and execution flow of mobile agents. Meanwhile, [16] presents a fault-tolerant protocol to ensure the execution of an agent monitoring its execution exactly once. In [18,19] the agent security is achieved distributing critical data and operations on mutually supporting agents, which migrate in disjoint host domains. Yee in [20] introduces a partial result based on authentication codes, extended later by Karjoth et al [21]. However, last works in mobile agent security [14,13] have broken this and other proposals [21,22,23,24], by means of "cut & paste" attacks and oracle exploits.

We can also find methods that protect itineraries. In [8] an itinerary protection method is introduced. The method is only suitable for sequence itineraries. Later, [10,25] present approaches with nested signatures and nested encryptions that protect the agent's route against attacks performed by the collusion of malicious contexts. Again, the approaches can not be applied on alternative or set itineraries. Afterwards, [9] introduces the concept of enhanced hash chains improving computational complexity of sequence itinerary protection protocols. And in [11] a set of different protection itinerary protocols are presented for each itinerary type. Unfortunately, protocols have different behaviours and computational complexities. Until now, complex itineraries with different recursive entries (a set itinerary inside an alternative itinerary) can not be protected.

In this paper, we improve previous solution making use of petri net modelling and cryptographic tools in order to protect flexible and complex itineraries from modifications in a unique general protocol.

4 Graphical Representation

Petri nets are well-known as a powerful modeling tool with a wide range of applications [26]. Their graphical representation of the modeled system can be used to reveal information about the system structure and dynamic behaviour.

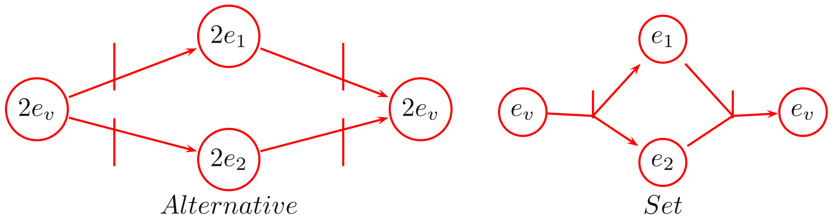
We assume that the reader has a basic understanding of Petri Nets. But, as a general reminder, we note that Petri Nets include three basic entities: place nodes (usually represented graphically by circles), transition nodes (represented graphically by solid bars) and edges that connect places to transitions or transitions to places. Furthermore, places can contain markers, called tokens, and tokens may move between place nodes by "firing" of the associated transitions. For a comprehensive introduction we refer the reader to [27].

Now, we can refine this model to characterize mobility in mobile agents. The set of locations to which a mobile agent is going to travel are place nodes, the transition nodes indicate the concurrency and each possible path is represented with an edge. The mobile agent is like a token that can be forked or joined in a transition node. The above presented formalism for mobile agent itineraries is graphically defined:

- A **Sequence** $[e_1, e_2]$ The m.a. must visit a list of entries in a defined order. We assume that a m.a. itinerary always begins and ends in a virtual node e_v. We usually associate the virtual nodes with the owner location, because he creates the m.a. and he will receive it with previous collected results.

$$\begin{tikzcd} e_v \arrow[r, "|" description] & e_1 \arrow[r, "|" description] & e_2 \arrow[r, "|" description] & e_v \end{tikzcd} \qquad (1)$$

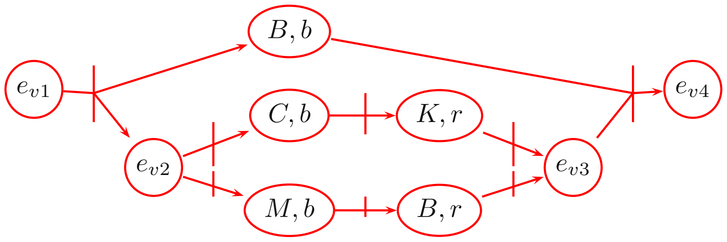
- An **Alternative** $< e_1, e_2 >$ Only one entry is visited from the whole set of entries. The mobile agent chooses a path and ends in the same location.
- A **Set** $\{e_1, e_2\}$ A list of entries is visited in any order. Using the mobile agent cloning property, the entries are concurrently visited. So, the transition node generates a set of edges in a fork phase and later join them.



(2)

Let see now, the graphical representation of a classic example from [16].

$$\begin{aligned}I = \{ & (BestFlowers, buyFlowers),\\ & < [\ (CentralTheatre, buyTicket), (KingsInn, reserveTable)],\\ & \quad [\ (ModernArts, buyTicket), (BeefHouse, reserveTable)] >\}\end{aligned}$$



(3)

The place nodes e_{vi} are virtual nodes and do not have any tasks to perform. They are used as union points for complex structures, where the m.a. will evaluate a condition or will be forked. In a real implementation, the virtual nodes are assigned to any trustworthy host. In order to improve the efficiency, we recommend to match the virtual nodes with the previous or posterior host. Thus, the m.a. doesn't need to migrate and communication costs are reduced.

Note that the use of virtual nodes is essential to evaluate complex structures. Using graphical notation we can observe where virtual nodes must be allocated in order to perform the protocol correctly without a malfunctioning of it.

5 General Protection Protocol

A flexible itinerary is composed of recursive entries of different types (sequence, alternative and set). The m.a. has to travel from one location to another one following the itinerary specification.

In order to analyze the itinerary, petri nets are used obtaining its graphical representation. New virtual nodes e_v arise, especially before and after fork and join transitions. Without loss of generality, we will always assume that first and last virtual nodes represent the owner. Remaining virtual nodes are assigned to a trustworthy hosts (owner inclusive). In these locations, the m.a. does not perform tasks but it could be forked in several child-agents, joined in a parent or a condition could be evaluated. The graphical representation is needed in order to evaluate efficiently complex itineraries. Also, the petri net representation clarifies the agent transitions that must be protected.

The protocol is focused on the digital envelope construction [28]:

$$digital\ envelope = P_j(r_i)\ \ E_{r_i}(Information) \tag{4}$$

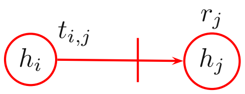
where r_i is the envelope's key and P_j is a public key. The owner of the private key can obtain the symmetric key r_i in order to obtain the information. The goal is that protected itinerary will be constructed as a chain of digital envelopes, in such a way that they only can be opened when the correct order is given.

Protection protocol:

1 Initialization Phase : The mobile agent's owner chooses a set of random uniformly values r_i with $1 \leq i \leq n$, where n is the number of place nodes of an itinerary (using its petri net representation). Then, he assigns one r_i to each node (the place nodes are marked from 1 to n).

2 For any transition $t_{i,j}$ it is performed:

– Normal transitions:



(5)

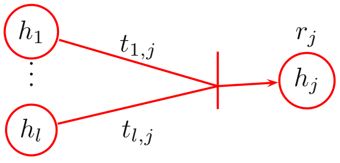
(a) Sign: The owner of the mobile agent $\mathcal{O}$ signs the information of the predecessor h_i and successor h_j host address, a trip marker t (see below), the random value r_j and an indicator of single transition '1'.

$$S_{\mathcal{O}}(h_i, h_j, t, r_j, 1) \tag{6}$$

(b) Public Encryption $P_j(\ )$: The mobile agent's owner encrypts the signed information using the public key of the host j. Then, he gets the tuple $t_{i,j}$ as a concatenation of the address of h_j and this encrypted value.

$$t_{i,j} = (h_j, P_j(S_{\mathcal{O}}(h_i, h_j, t, r_j, 1))) \tag{7}$$

– l-Join transitions:



(8)

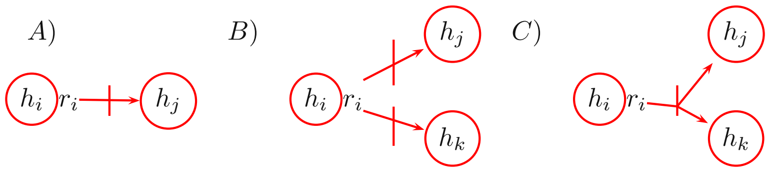
(a) Sign: In l-join transitions $t_{1,j}, t_{2,j}, \cdots, t_{l,j}$ (see 8)the owner chooses an extra random value $r_{k,j}$ for each edge, where $1 \leq k \leq l$, such that $\bigotimes_{k=1}^{l} r_{k,j} = r_j$, where $\otimes$ is xor operation. Therefore, the signed information will be composed with these values and with an indicator of l-join transition 'l'.

$$S_{\mathcal{O}}(h_1, h_j, t, r_{1,j}, l), \cdots, S_{\mathcal{O}}(h_m, h_j, t, r_{m,j}, l) \tag{9}$$

(b) Public Encryption $P_x(\ )$: The owner of the mobile agent encrypts the signed information as in a normal transition.

$$t_{1,j} = (h_j, P_j(S_{\mathcal{O}}(h_1, h_j, t, r_{1,j}, l))), \cdots,$$
$$t_{l,j} = (h_j, P_j(S_{\mathcal{O}}(h_l, h_j, t, r_{l,j}, l)))$$

3 Symmetric Encryption $E_x(\ )$: The owner encrypts the private information of each node with the out-going transitions using the random value r_i fixed at step 1. In a sequence itinerary the private information is composed by the method m_i or task to be performed (case A). However, in an alternative (case B) or set (case C) itinerary there is also a condition to be evaluated or a fork action to be performed. So, we note



(10)

$$A\ )\ e_i = E_{r_i}(m_i, -, t_{i,j})$$
$$B\ )\ e_i = E_{r_i}(m_i, cond, t_{i,j}, t_{i,k})$$
$$C\ )\ e_i = E_{r_i}(m_i, fork, t_{i,j}, t_{i,k})$$

4 End : Once the protocol has been employed over the petri net representation, a new protected itinerary $I = (e_1, e_2, \cdots, e_n)$ is obtained. In order to protect the integrity the owner signs a hash function of itinerary and annex it to m.a.

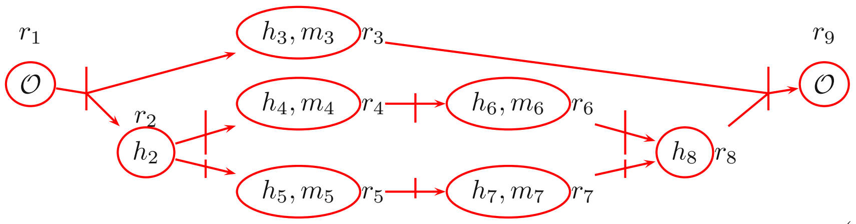
$$S_{\mathcal{O}}(hash(t, I)) = S_{\mathcal{O}}(hash(t, e_1, e_2, \cdots, e_n)). \tag{11}$$

It is easy to see that we have a list of entries. All are encrypted with a random symmetric key r_i and it is not feasible to obtain any information about it meanwhile the security depends on the security of the used cryptosystem. Moreover, we can mix the entries in order to hide the real position inside the itinerary. So, nobody would know how many entries leave until the mobile agent returns to the owner or how many task the mobile agent is going to perform because there are virtual nodes.

Initially, each node has to decrypt all entries to find his own entry. In order to reduce this computational cost, we could replace each entry e_i by a tuple $(h(r_i), e_i)$. Thus, the node can find the right entry directly.

5.1 Example

Working with previous example, we can apply the protocol over the graphical representation (see figure (3)), obtaining the protected itinerary I:



(12)

$$\begin{aligned} I = (\ & E_{r_1}(-, fork, t_{1,2}, t_{1,3}), E_{r_2}(-, cond, t_{2,4}, t_{2,5}), E_{r_3}(m_3, -, t_{3,\mathcal{O}}), \\ & E_{r_4}(m_4, -, t_{4,6}), E_{r_5}(m_5, -, t_{5,7}), E_{r_6}(m_6, -, t_{6,8}), \\ & E_{r_7}(m_7, -, t_{7,8}), E_{r_8}(-, -, t_{8,\mathcal{O}}), S_{\mathcal{O}}(hash(t, I))) \end{aligned}$$

5.2 Mobile Agent Migration

The m.a. migrates following the itinerary between hosts that offer environments in which the code can be executed. The owner is associated with the first place node h_1 and knows the random key r_1. So, when he opens the first digital envelope $E_{r_1}(-, -, t_{1,j})$ and obtains the transition $t_{1,j}$. Any mobile agent system h_j will be able to receive the m.a. with the protected itinerary and the associated transition $t_{1,j}$. We assume that hosts use a non-repudiation communication protocol in order to communicate among them [29] [8].

5.3 Verification Protocol

Any host h_j will receive the mobile agent with the protected itinerary and the associated transition $t_{i,j}$. The transition indicates the destination host address h_j and a private information for that host.

Verification protocol:

1 Recover private information : The h_j mobile agent system receives the agent and uses its private key to decrypt the private data.

$$t_{i,j} = (h_j, P_j(S_{\mathcal{O}}(h_i, h_j, t, r_j, l))) \ \Rightarrow \ S_{\mathcal{O}}(h_i, h_j, t, r_j, l). \tag{13}$$

2 Integrity check : The host will check the owner's digital signature $S_{\mathcal{O}}()$ in order to detect modifications. Also, it will obtain the agent's source address h_i, his own address h_j, the trip marker t, the random value r_j and the indicator of single or multiple transition l.

3 Source Verification : Communications between hosts are not anonymous. So, with the previous host address h_i signed, every host is able to check the agent's source. If the check fails, the agent should be returned to the owner.

4 Verifiability : Using the host address h_j supplied in the transition (13), the mobile agent system can detect if it is part of the initial route.

5 Exactly once: The trip marker t identifies the agent's journey and prevents replay and cut-&-paste attacks. Without this value, a malicious host could change the remaining itinerary of the agent with an old mobile agent itinerary sent by the same owner. The trip marker is composed of a timestamp and a hash function of the mobile agent's code:

$$t = timestamp \parallel hash(\text{mobile agent's code}) \tag{14}$$

Timestamp is used in order to guarantee a unique trip marker while the hash of the mobile agent's code links the trip marker with the mobile agent instance. The visited host should store the trip marker in a database, at least as long as the owner signature is valid plus a safety margin for clock skew.

An attacking host can not reuse neither modify any entry. Entries are protected with symmetric encryption and contains the trip marker, that is checked by the agency.

6 Recovering the digital envelope key: If the mobile agent system receives an indicator of single transition, then r_j will be the digital envelope key.

Otherwise, it will have received a random value $r_{k,j}$, where $1 \leq k \leq l$. The mobile agent system will have to wait for other $l-1$ child mobile agents in order to recover the real digital envelope key, computing:

$$\bigotimes_{k=1}^{l} r_{k,j} = r_j$$

where $\otimes$ is xor operation.

7 Forward Privacy : The host checks the list of entries looking for his own entry e_j. Using the digital envelope key r_j recovered in the previous step, the server will retrieve the method m_j to perform. Note that the digital envelope key is available only if previous entries have been visited, in such a way forward privacy is guaranteed.

$$e_j = E_{r_j}(m_j, cond, t_{j,x}, t_{j,w}) \quad \Rightarrow \quad m_j, cond, t_{j,x}, t_{j,w} \tag{15}$$

The host can find new transitions $t_{j,x}, t_{j,w}$ where mobile agent has to migrate. This is the minimal information needed by the host. Remaining itinerary information is encrypted to prevent itinerary analysis by an adversary.

Sometimes, the mobile agent system can find a condition or a fork action with a set of transitions, indicating an alternative or set itinerary, respectively. When the host finds a condition *cond*, it has to evaluate it in order to choose next route. Otherwise, the mobile system will fork the mobile agent parent in several mobile agent children (one for each transition).

5.4 Complexity of the Protocol

The complexity of the protection protocol can be computed as follows. In step 2, assuming all transitions are l-join transitions (worst case), the number of cryptographic operations is $O(2ln)$, where n is the number of place nodes and 2 indicates the digital signature and the public encryption. In step 3, n digital envelopes are created for each entry, so it requires an additional $O(n)$ cryptographic operations. The complexity of the verification protocol is $O(ln)$, where ln is the number of decryptions of step 1, assuming all transitions are l-join transitions (worst case).

Summing up, the complexity of protection and verification protocols is $O((3l+1)n)$ cryptographic operations.

6 Conclusions

In this paper we have presented a new protocol to protect general flexible prefixed itineraries of mobile agents. This protocol uses petri net modeling, digital signatures, symmetric and public encryption schemes, and a p.k.i.

Although itinerary protection protocols are not new, we present a complete solution protecting any possible itinerary, using a unique protocol and improving previous versions with new security features. Our protocol reduces the route information provided to visited hosts and it provides strong protection against tampering and itinerary analysis attacks. Up to now, previous works were ad hoc solutions to each kind of itinerary, while our protocol is generic.

Acknowledgments

This work has been partially funded by the Spanish Government Commission CICYT (TIC2000-0232-P4), Catalan Government Department DURSI (2001SGR 00219), and the company Camper S.A.U.

References

1. D. Chess, B. Grosof, C. Harrison, D. Levine, C. Parris, and G. Tsudik. Itinerant agents for mobile computing. In *IEEE PCM, 2*, pages 34–49, October 1995.
2. Odyssey: Beta Release 1.0. Available as part of the Odyssey package at, 1997. `http://www.genmagic.com/agents/`.
3. Voyager technical overview. Technical report, ObjectSpace White Paper, 1997. `http://www.objectspace.com/voyager`.
4. D. Wong, N. Paciorek, T. Walsh, J. DiCelie, M. Young, and B. Peet. Concordia: An Infrastructure of Collaborating Mobile Agents. LNCS 1219, 86–97. SV. 97.
5. N. Karnik. *Security in Mobile Agent Systems.* PhD thesis, Department of Computer Science and Engineering, University of Minnesota, 1998.
6. Danny B. Lange and Mitsuru Oshima. *The Aglet cook-book.* In progress. `http://www.trl.ibm.co.jp/aglets/aglet-book/index.html`.

7. V. Roth and V. Conan. Encrypting Java Archives and its application to mobile agent security. In Frank Dignum and Carles Sierra, editors, *Agent Mediated Electronic Commerce: A European Perspective*, *LNAI 1991*, pages 232–244. SV, 2001.
8. J. Borrell, S. Robles, J. Serra, and A. Riera. Securing the Itinerary of Mobile Agents through a Non-Repudiation Protocol. In *IEEE International Carnahan Conference on Security Technology*, pages 461–464, 1999.
9. J. Domingo and J Herrera. Enhanced Hash Chains for Efficient Agent Route Protection. Tech report, Submitted to Information Processing Letters, April 2000.
10. D. Westhoff, M. Schneider, C. Unger, and F. Kaderali. Methods for protecting a mobile agent's route. In *ISW'99*, LNCS 1729, pages 51–71. SV, Nov 1999.
11. J. Borrell and J. Mir. Protecting General Flexible Itineraries of Mobile Agents. LNCS 2288, pages 425–440, Seul, Corea, SV 2002.
12. Volker Roth. Mutual protection of co–operating agents. In Jan Vitek and Christian Jensen, editors, *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, vol 1603 of *LNCS*, pages 275–285. SV, NY, USA, 1999.
13. Volker Roth. On the robustness of some cryptographic protocols for mobile agent protection. In *Proc. Mobile Agents 2001*, *LNCS 2240*. SV, Dec 2001.
14. V. Roth. Programming Satan's agents. In Klaus Fischer and Dieter Hutter, editors, *SEMAS 2001*, *ENTCS 63*. Elsevier. Science Publishers, 2002. ISBN 0444512268.
15. S. Robles, J. Mir, J. Ametller, and J. Borrell. Implementation of secure Architectures for Mobile Agents in MARISM-A. In *Proceeding of 4th. IW on Mobile Agents for Telecommunication Applications*, LNCS 2521, pages 182–191, 2002.
16. M. Straßer, K. Rothermel, and C. Maifer. Providing Reliable Agents for Electronic Commerce. LNCS 1402, pages 241–253. SV, 1998.
17. G. Vigna. Mobile Agents and Security. In *Mobile Agents and Security*, LNCS 1419, pages 137–153. SV, 1998.
18. Volker Roth. Secure recording of itineraries through cooperating agents. In *Proc. 4th ECOOP*, pages 147–154, July 1998. Dépot légal 010598/150.
19. V. Roth. Mutual protection of co-operating agents. LNCS 1603, p 257–285. SV'99.
20. B. Yee. A Sanctuary for Mobile Agents. In *DARPA Workshop on Foundations for Secure Mobile Code Workshop*, March 1997.
21. G. Karjoth, N. Asokan, and C. Gülcü. Protecting the Computation of Free-Roaming Agents. In *Mobile Agents*, LNCS 1477, pages 194–207. SV, 1998.
22. Gunter Karjoth. Secure Mobile Agent-based merchant brokering in distributed marketplaces. In *Proceedings AS/MA 2000*, LNCS 1882, pages 44–56. SV, 2000.
23. A. Corradi, R. Montanari, and C. Stefanelli. Mobile agent protection in internet environment. (COMPSAC'99), pages 80–85, 1999.
24. N. M. Karnik and A. R. Tripathi. Security in the Ajanta Mobile Agent System. Technical report, Department of Computer Science, U. of Minnesota, May 1999.
25. D. Westhoff, M. Schneider, C. Unger, and F. Kaderali. Protecting a Mobile Agent's Route Against Collusions. In *SAC'99*, LNCS 1758, pages 215–225. SV, 2000.
26. T. Murata. *Petri Nets: Properties, Analysis and Applications*, vol 77. IEEE, 89.
27. J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, Inc., Englewood Cliffs, 1981.
28. PCKS7. Cryptographic Message Syntax Standard. Technical report, An RSA Laboratories Technical Note, November 1993.
29. Jianying Zhou. *Non-repudiation in Electronic Commerce*. Computer Security Series. Arthech House, 2001.

Implementation and Performance Evaluation of a Protocol for Detecting Suspicious Hosts

Oscar Esparza, Miguel Soriano, Jose L. Muñoz, and J. Forné

Department of Telematics Engineering. Technical University of Catalonia.
C/ Jordi Girona 1 i 3. Campus Nord, Mod C3, UPC. 08034 Barcelona. Spain.

{oscar.esparza, soriano, jose.munoz, jforne}@entel.upc.es

Abstract. Mobile agents are software entities that consist of code, data and state, and that can migrate autonomously from host to host performing some actions on behalf of a user. Security issues restrict the use of code mobility despite its benefits. The protection of mobile agents from the attacks of malicious hosts is considered by far the most difficult security problem to solve in mobile agent systems. In this paper the authors present the implementation and the performance evaluation of a new protocol for detecting suspicious hosts in the mobile agent environment. The protocol is based on limiting the execution time of mobile agents in the executing hosts and was previously presented in [6]. The authors also present how this protocol can improve the usability of the cryptographic traces approach [12].

1 Introduction

Mobile agents are software entities that consist of code, data and state, and that can migrate from host to host performing actions autonomously on behalf of a user. Therefore, mobile agents are especially useful to perform functions automatically in almost all electronic services. Unfortunately, massive use of mobile agents is restricted by security issues despite their benefits.

There are two prior entities involved in this scenario: the mobile agent and the host. Protection is necessary when trustworthy relationships between entities cannot be assured. Host protection from malicious agent attacks can be achieved by limiting the execution environment with sand-boxing techniques and a proper access control. Communication security is based on well-known cryptographic protocols. Nevertheless, there is no proper solution to protect mobile agents completely against the attacks of malicious hosts.

In [6] the authors presented a protocol that detects hosts that are suspicious of malicious behavior. The protocol measures the agent's execution times in the hosts, and these times are the evidence of the good or the bad behavior of the executing hosts. In this paper we present the implementation and some performance evaluation results of this protocol. Additionally, we present how the protocol can improve the usability of the cryptographic traces approach [12].

The rest of the paper is organized as follows: Section 2 presents the state-of-the-art related solutions; Section 3 presents how to use the protocol with the cryptographic traces; Section 4 presents some implementation details and some performance evaluation; finally, some conclusions can be found in Section 5.

2 Malicious Hosts

The attacks performed to a mobile agent by an executing host are considered the most difficult problem regarding mobile agent security. Assuring the integrity and authentication of code, data or results that come from other hosts is possible by using digital signature or encryption techniques. However, it is difficult to detect or prevent the attacks performed by a malicious host during the agent's execution. Malicious hosts could try to get some profit of the agent reading or modifying the code, the data, the communications or even the results due to their complete control on the execution.

There are some published approaches that deal with the problem of malicious hosts, but none of them solves it completely. Yee introduces the idea of a "Sanctuary" [13] as a closed tamper-proof hardware subsystem where agents can be executed in a secure way, but this forces each host to buy a hardware equipment and to consider the hardware provider as trusted. The environmental key generation [9] makes the agent's code impossible to decrypt until the proper conditions happen on the environment, so previous analysis from hosts is avoided. However, there is no protection once the code has been decrypted. Hohl presents obfuscation [8] as the mechanism to assure the execution integrity during a period of time, but this time depends on the capacity of the malicious host to analyze the obfuscated code. The use of encrypted programs [11] is proposed as the only way to give privacy and integrity to mobile code. Further improvements in this sense were addressed in [5,4]. The difficulty here is to find functions that can be executed in an encrypted way. Vigna [12] introduced the idea of cryptographic traces that the agent takes during execution. The agent sender asks for the traces if it wants to verify execution, but verification is only performed in case of suspicion. Roth [10] presents the idea of cooperative agents that share secrets and decisions and have a disjunct itinerary. This fact makes collusion attacks difficult, but not impossible.

3 Usability of the Protocol

In [6] the authors presented a protocol that detects hosts that are suspicious of malicious behavior. As malicious hosts need time to analyze and modify a mobile agent in order to take some profit, we can detect suspicious behaviors by controlling the time spent executing the agent. Each executing host counts the running time with regard to a time reference. When the agent reaches the origin host, a set of time checks evaluates if the hosts spent more time than expected executing the agent. A host is suspicious if it does not pass any of the time checks. The protocol also protects the mobile agent while it is migrating from host to host by using cryptographic techniques.

In this section we present how this protocol can be used jointly with the cryptographic traces approach [12]. Before this, a short overview of the protocol must be introduced. As it is said, further details about the message and agent passing of the protocol can be found in [6].

3.1 Overview of the Protocol

These are briefly the steps that the entities involved in the protocol must follow:

1. The Home (origin host) assumes the role of client and sends a message to each executing host with the following values:
   - A time reference counter that will be used as the clock of the system.
   - A session key that will be used to encrypt the mobile agent with a symmetric algorithm.
2. Acting as a server, each host receives the message and starts the time reference counter.
3. Home encrypts the agent's code and data using the session key to protect them during the journey across the network and sends the agent to the first host in the itinerary.
4. Each host in the agent's itinerary must perform the following tasks:
   (a) It takes the agent's arrival time regarding the time reference counter.
   (b) It decrypts the agent by using the session key.
   (c) It executes the agent's code.
   (d) It saves the finalization time regarding the time reference counter.
   (e) It encrypts the results and the arrival and finalization times with the public key of the Home.
   (f) Finally, it sends the agent to the following host in the itinerary.
5. The Home verifies the temporal correctness of the execution when the agent returns. The following time checks are performed:
   - A host is suspicious of malicious actions if its execution time exceeds the expected execution time.
   - The finalization time of a host cannot be greater than the arrival time of the next host.
   - The difference between the finalization time of a host and the arrival time of the next host must be similar to the estimated transmission time.

   A host is considered suspicious if it does not pass any of the time checks.

3.2 Drawbacks

These drawbacks limit the environments where we can use the protocol:

- The last executing host must send back the agent to the origin host because the agent carries the arrival and finalization times. Without these times the temporal correctness cannot be verified.

- The origin host must estimate the execution times in the hosts and some transmission times between hosts. How these times are estimated is out of the scope of this paper. In fact, we assume that these tasks have been delegated to external entities that can use several policies. The easiest policy is to consider deterministic and fixed times, but sophisticated algorithms can be found depending on CPU, memory available, network bandwidth, propagation delays or other parameters.
- The protocol can detect honest hosts as suspicious if they spend more time than expected during execution. In this sense, the origin host cannot punish the suspicious hosts because there are no proofs of their malicious behavior.
- The synchronization mechanism of the protocol can be easily improved by using for example the Network Time Protocol in all the hosts or any other way to get a system time reference.

Because of all these drawbacks, we recommend using the suspicious detection protocol for services in which (1) the mobile agents return to the origin host, for instance agents that carry results of the execution; (2) there must be the possibility of estimating the execution time in the hosts. We discourage using the protocol with agents that have dependencies with other agents or that must compute large amounts of data, because these facts make the estimation difficult; (3) Additionally, we consider that the protocol must be used jointly with an approach that proves the malicious behavior of the hosts. The rest of the section presents how the protocol can improve the capabilities of the cryptographic traces approach [12].

3.3 Integration with the Cryptographic Traces Approach

The prevention of attacks performed by a malicious host that is running the agent seems a difficult problem that does not have an appropriate solution. In the cryptographic traces approach [12], Vigna considers that detection of attacks is the best way to protect mobile agents against the attacks of the malicious hosts. In his proposal, the running agent takes traces of instructions that alter the agent's state due to external variables. The host sends a hash of the traces with the results because the complete traces are too large. If the agent owner wants to verify execution, it asks for the traces and executes the agent again. If new execution does not agree with the traces, the host is cheating. The approach not only detects attacks, but it also proves the malicious behavior of the host. This approach has various drawbacks: (1) Verification is only performed in case of suspicion, but how a host becomes suspicious is not explained; (2) Each host must store the traces for an indefinite period of time because the origin host can ask for them; and (3) a Third Trusted Party (TTP from here on) is needed in order to punish malicious behaviors.

The suspicious detection protocol solves the first two drawbacks of the Vigna's approach:

- There is a way to know which hosts are suspicious, i.e. those hosts that spent more time than expected executing the agent.

– As the origin host knows who are the suspicious hosts just when the agent returns, the traces can be asked directly to those suspicious hosts, and consequently the executing hosts do not have to store the traces for an indefinite period of time.

In conclusion, the cryptographic traces can be used with the suspicious detection protocol to achieve an effective attack detection and proving mechanism.

The lack of a TTP with punishment capabilities can also be solved by adding a Host Revocation Authority (HoRA) to the mobile agent system [7]. The HoRA controls which are the hosts that acted maliciously and for this reason they have been revoked. The origin host must consult the revocation information before sending an agent in order to remove from the agent's itinerary all the revoked hosts. As a consequence, revoked hosts will not receive mobile agents any more.

4 Implementation and Performance Evaluation

In this paper we also present the implementation and some performance evaluation results of the suspicious detection protocol. For the test we used mobile agents that performed easy tasks because a complicated agent could distort the results. In particular, the mobile agents only have to find the more economic flight to a particular destination in a table located in each host.

4.1 Software Tools and Hardware Specifications

The programming language used in the implementation of the protocol is Java, and more precisely, the Aglet Software Development Kit 2.0.1 (ASDK or Aglets Workbench) [3] designed by the IBM Tokyo Research Laboratory. ASDK provides an API to create aglets (AGent-appLETS), that is a Java object that can be used as a mobile agent. ASDK also gives a graphic environment called Tahiti, where the aglets can be sent, received and executed. This mobile agent server, Tahiti, is supported by a Java Virtual Machine (JVM) included in the Java Development Kit (JDK) 1.3.1 [2]. The Java Cryptography Extension (JCE 3.01) [1] includes some additional cryptographic Java libraries that have been used in order to protect the mobile agent during its journey across the network.

The results have been taken in a three computer laboratory, one computer working as Home and two working as hosts. Host1 and Host2 are Pentium II at 400 MHz, 64 MB of RAM and Linux SuSE 8.1 as OS. Home was a laptop with a 1GHz processor, 256 MB of RAM and Windows XP as OS. An Ethernet Hub working at 10 Mbps connects the three computers.

4.2 Results

The implementation of the protocol has the possibility of using encryption tools to protect the agent and the information that carries during its journey. Obviously, the performance of the protocol depends on the way we use this encryption tools, and specially the key lengths of the cryptographic protocols. This implementation has the following operational modes:

- The origin host sends the mobile agent without any kind of security tool. In the results we denote this mode with a key length=-50. Obviously this value has no sense, but we include the results of this mode in order to compare them with the results of the following mode.
- The origin host uses the protocol, but without using any kind of cryptographic tool, so this mode does not assure the privacy of the information. Comparing these results with the results of the previous mode we can see how verifying the temporal correctness of the execution affects to the performance of the system. In the results we denote this mode with a key length=0.
- The origin host uses not only the protocol, but also some cryptographic tools to perform authentication, privacy and integrity of the information.

These are the encryption techniques that we have used:

- Symmetric encryption: all the hosts share the same 64 bit DES [1] or 128 bit Triple DES session key to protect the agent during its journey. The lifetime of this key is short because a new session key is generated each time an agent is sent.
- Public key encryption: we use public key encryption to protect the results of the execution and the arrival and finalization times. Additionally, all the messages must be properly signed. For this reason, we need a Public Key Infrastructure (PKI) in our system. In particular, we use RSA with 512 bit, 1024 bit and 2048 bit key lengths.

The rest of the section presents some performance evaluation results that show how the protocol affects in some relevant aspects to the hosts and the network.

Execution Time in the Host Figure 1 shows the average time needed by a host to execute the agent. This time depends on the DES, Triple DES and RSA key lengths. The picture shows the following measurements: (1) execution time in the host without any kind of security (RSA Key=-50); (2) execution time in the host using the protocol, but without using any kind of cryptographic tool (RSA Key=0); (3) other execution times in the host depending on the DES, Triple DES and RSA key lengths. These measurements were taken with a 50 % of CPU load in the server.

From the figure we can extract that the implemented protocol does not suppose a great increase on the execution time of the host, as the time without the protocol (30 ms) is similar to the time with the protocol and without cryptography (52 ms). Obviously, the use of cryptographic techniques to protect the agent and its results increases the time spent in the host.

[1] The use of the 64 bit DES key is not recommended because it is considered too weak for actual services. Despite this fact, this possibility has been included in the implementation in order to underline that the performance results are almost equal for both DES and Triple DES algorithms, but Triple DES is much more secure than DES.

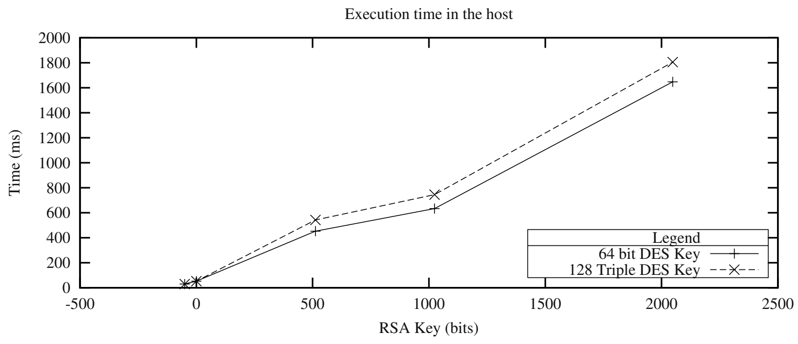


Fig. 1. Execution Time in the Host

Execution Time in the Home Figure 2 shows the average time needed by the Home to recover the agent's results and to verify if there are suspicious hosts. From the figure we extract that the time checks do not suppose a great increase on the execution time of the Home, as the without-security time (0,1 s) is almost equal to the time with the protocol and without cryptography (0,15 s).

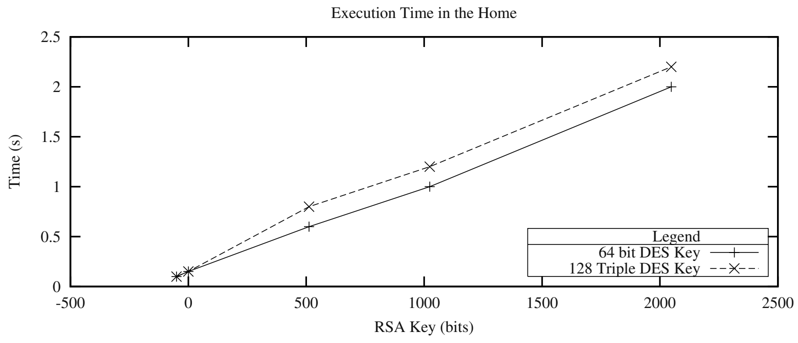


Fig. 2. Execution Time in the Home

Total Search Time Figure 3 shows the total time spent by the agent during its journey, i.e. from the agent is sent to it is received. In these tests the network load was of 30%, and the CPU load in the host was around a 50%. In these measurements are included the effects of serializing and de-serializing the agent in each host.

CPU Load in the Host Figure 4 shows the average CPU load during the execution time in the host. From the figure we can extract that the protocol

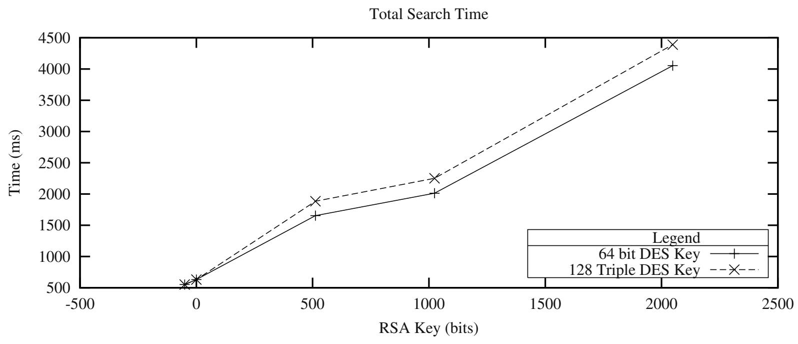


Fig. 3. Time Spent for the Agent during its Journey

does not suppose a great increase on the CPU load, as the load without security (5%) is similar to the load with the protocol and without cryptography (9%).

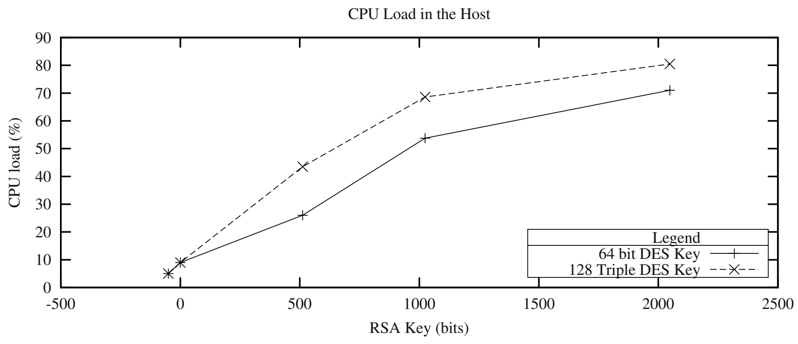


Fig. 4. CPU Load in the Host

CPU Load in the Home Figure 5 shows the average CPU load when the Home tries to recover the agent's results and it verifies if there are suspicious hosts. From the figure we can extract that the time checks do not suppose a great increase on the CPU load of the Home, as the load without security (10%) is similar to the load with the protocol and without cryptography (14%).

Agent Size Figure 6 shows the agent's size in bytes. These measurements give us an idea of how the network load is increased by the protocol. From the figure we can extract that the without-security size is almost equal to the size using the protocol, but without cryptography. Obviously, the agent size is increased on

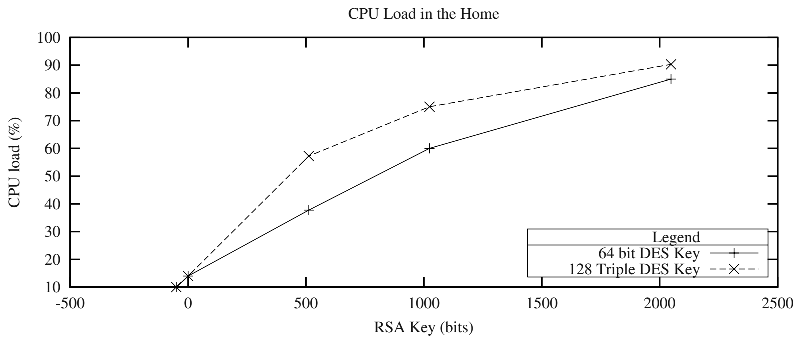


Fig. 5. CPU Load in the Home

each host that the agent traverse, as the results and the arrival and finalization times must be added to the agent. We used in this case a 128 Triple DES key.

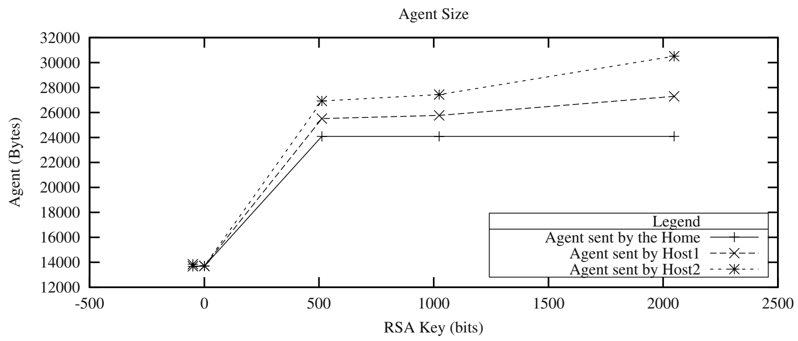


Fig. 6. Agent Size

5 Conclusions

In this paper, the authors present the way to use the new suspicious detection protocol to improve the usability of the cryptographic traces approach [12]. Using jointly both schemes, a complete and effective attack detection and proving mechanism is achieved. Some drawbacks of the protocol limit its use to environments in which (1) the mobile agents return to the origin host; and (2) there must be the possibility of estimating the execution time in the hosts.

Additionally, the implementation and some performance evaluation results of the protocol have been presented. Accordingly to the results, verifying the temporal correctness of the agent's execution does not suppose a significative increase in the performance of the hosts and the network. If privacy of the information must be assured, we recommend protecting the agent at least with a 128 bit Triple DES key and a 1024 bit RSA key.

References

1. Java Cryptography Extension (JCE). Institute for Applied Information Processing and Communication of the Graf University of Tecnology. http://jce.iaik.tugraz.at/download/evaluation/index.php.
2. Java Development Kit. http://java.sun.com/downloads/.
3. SourceForge projects, Aglet Software Development Kit (ASDK). http://wwww.sourceforge.net/projects/aglets.
4. J. Algesheimer, C. Cachin, J. Camenisch, and G. Karjoth. Cryptographic security for mobile code. In *IEEE Symposium on Security and Privacy*, 2001.
5. C. Cachin, J. Camenisch, J. Kilian, and Joy Müller. One-round secure computation and secure autonomous mobile agents. In *27th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 1853 of *LNCS*. Springer-Verlag, 2000.
6. O. Esparza, M. Soriano, J.L. Muñoz, and J. Forné. A protocol for detecting malicious hosts based on limiting the execution time of mobile agents. In *IEEE Symposium on Computers and Communications - ISCC'2003*, 2003.
7. O. Esparza, M. Soriano, J.L. Muñoz, and J. Forné. Host Revocation Authority: a Way of Protecting Mobile Agents from Malicious Hosts. In *International Conference on Web Engineering (ICWE 2003)*, volume 2722 of *LNCS*. Springer-Verlag, 2003.
8. F. Hohl. Time Limited Blackbox Security: Protecting Mobile Agents From Malicious Hosts. In *Mobile Agents and Security*, volume 1419 of *LNCS*. Springer-Verlag, 1998.
9. J. Riordan and B. Schneier. Environmental Key Generation Towards Clueless Agents. In *Mobile Agents and Security*, volume 1419 of *LNCS*. Springer-Verlag, 1998.
10. V. Roth. Mutual protection of cooperating agents. In *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, volume 1906 of *LNCS*. Springer-Verlag, 1999.
11. T. Sander and C.F. Tschudin. Protecting mobile agents against malicious hosts. In *Mobile Agents and Security*, volume 1419 of *LNCS*. Springer-Verlag, 1998.
12. G. Vigna. Cryptographic traces for mobile agents. In *Mobile Agents and Security*, volume 1419 of *LNCS*. Springer-Verlag, 1998.
13. B.S. Yee. A sanctuary for mobile agents. In *DARPA workshop on foundations for secure mobile code*, 1997.

Author Index